Discrete Applied Mathematics 17 (1987) 67–105
North-Holland



# AN OPTIMAL TIME ALGORITHM FOR THE $k$-VERTEX-CONNECTIVITY UNWEIGHTED AUGMENTATION PROBLEM FOR ROOTED DIRECTED TREES

Toshimitsu MASUZAWA, Ken'ichi HAGIHARA and Nobuki TOKURA
*Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University, 1-1 Machikaneyama, Toyonaka 560, Japan*

Received April 1985
Revised November 1985

For a given digraph $G=(V,A)$ and a positive integer $k$, the $k$-vertex-connectivity unweighted augmentation problem ($k$-VCUAP) for $G$ is to find a minimum set of arcs $A'$ ($A' \subseteq (V \times V - A)$) such that the digraph $(V, A \cup A')$ is $k$-vertex-connected. It is known that the time-complexity of 1-VCUAP for every digraph is $\theta(|V|+|A|)$. However, it remains still open whether or not there exist polynomial time algorithms for $k$-VCUAP's ($k \geq 2$) for digraphs. This paper shows that the time-complexity of $k$-VCUAP ($k \geq 2$) is $\theta(k|V|)$ for every rooted directed tree.

## 1. Introduction

The *k-vertex-connectivity augmentation problem* ($k$-VCAP) for a graph $G=(V,E)$ and a cost function on $V \times V$, is the one to find a minimum cost set of edges $E'$ such that $G=(V,E \cup E')$ is $k$-vertex-connected. There are some other related problems; the one for a digraph, the one for $k$-edge-connectivity and so on. These problems are discussed in [3], [5] and [7]–[17].

The *k-vertex-connectivity unweighted augmentation problem* ($k$-VCUAP) is $k$-VCAP with a constant cost function. This paper considers $k$-VCUAP for rooted directed trees.

The following results about $k$-VCAP and $k$-VCUAP for digraphs are known (Table 1):

(1) 1-VCAP for digraphs is NP-complete even for acyclic digraphs with 1-2 valued cost functions [5].

(2) The time-complexity of 1-VCUAP for every digraph $G=(V,A)$ is $\theta(|V|+|A|)$ [3].

(3) The time-complexities of 2-VCUAP and 3-VCUAP for every rooted directed tree $G=(V,A)$ are $\theta(|V|)$ [12].

(4) The time-complexity of $k$-VCUAP ($k \geq 4$) for every ternary tree $G=(V,A)$ is $\theta(k|V|)$ [13].

(5) The time-complexity of $k$-VCUAP ($k \geq 2$) for every digraph $G=(V,A)$ whose underlying graph is a tree such that the degree of each vertex is just $k$ is $\theta(k|V|)$ [12].

Table 1. Known results of $k$-VCAP and $k$-VCUAP for digraphs

| | $k$-VCAP | $k$-VCUAP |
|---|---|---|
| $k=1$ | NP-complete [5]<br>acyclic digraphs with<br>1-2 valued cost functions | $\theta(\|V\|+\|E\|)$ [3] |
| $k\geq 2$ | open | $(k=2,3)$ rooted directed trees<br>$\theta(\|V\|)$ [12] |
| | | $(k\geq 4)$ ternary trees<br>$\theta(k\|V\|)$ [13] |
| | | digraphs whose underlying graphs are trees such that degree of each vertex is $k$<br>$\theta(k\|V\|)$ [12] |
| | | $(k\geq 4)$ digraphs whose underlying graphs are trees such that degree of each vertex is at most three<br>$O(k\|V\|)$<br>(approximate solution) [12] |

(6) For $k$-VCUAP ($k\geq 4$) for every digraph $G=(V,A)$ whose underlying graph is a tree such that the degree of each vertex is at most 3, an approximate solution including at most one more arc than the optimal solution is found in $O(k|V|)$ time [12].

However, it remains still open whether or not there exist polynomial time algorithms for $k$-VCUAP ($k\geq 2$) for digraphs.

This paper shows an algorithm for solving $k$-VCUAP ($k\geq 2$) for every rooted directed tree $G=(V,A)$ with $O(k|V|)$ time. Also, it shows that the problem requires $\Omega(k|V|)$ time. This means that the running time of our algorithm is optimal within a constant factor.

The key-idea of our algorithm is as follows:

We introduce two main concepts:

(i) the class of $k$-regular and $k$-connected digraphs called super daisy chains, and

(ii) two operations on $k$-regular and $k$-connected digraphs to construct a digraph with $k$-regularity and $k$-connectivity preserved.

The digraph class and the two operations mentioned above make it possible to augment any $k$-ary tree to $k$-regular and $k$-connected by taking advantage of the divide-and-conquer principle.

By the extension of the above method for $k$-ary trees, our efficient algorithm to solve $k$-VCUAP ($k\geq 2$) for every rooted directed tree is attained.

## 2. Definitions, notations and propositions

This section introduces terminology and notations. Most of them are standard but some are newly introduced in this paper.

**Notation 1.** $\langle a_1, a_2, \ldots, a_p \rangle$: the *sequence* of $a_1, a_2, \ldots, a_p$ in this order. $p$ is called the *length of the sequence*.
For a finite set $A$,
$|A|$: the number of elements in $A$,
$\langle\langle A \rangle\rangle$: any sequence of length $|A|$ which consists of all elements in $A$,
$\langle\langle A \rangle\rangle_i$: the $i$-th element $(1 \leq i \leq |A|)$ of $\langle\langle A \rangle\rangle$.
For any integers $m$ and $n$ $(m \leq n)$,
$[m \cdots n] = \{m, \ldots, n\}$: the set of integers such that $i \in [m \cdots n]$ if and only if $m \leq i \leq n$.

**Definition 1.** A *digraph* $G=(V, A)$ is specified by a finite set $V$ of *vertices* and a finite set $A$ of *arcs* which are ordered pairs of distinct vertices.

In this paper, it is assumed that a digraph has neither self-loops (arc of the form $(u, u)$) nor parallel arcs. This assumption is useful to simplify the following discussions without any loss of generality: the solution of $k$-VCUAP for $G$ with self-loops and/or parallel arcs could be obtained as the solution for the reduced digraph with self-loops deleted and parallel-arcs replaced by an arc.

Throughout this paper, $m, n, i, j$ and $k$ will denote positive integers and $G=(V, A)$ will denote a digraph unless otherwise stated.

**Notation 2.** For $G=(V, A)$,
$V(G)$: the vertex set of $G$; that is, $V(G)=V$,
$A(G)$: the arc set of $G$; that is, $A(G)=A$.
For an arc $e=(u, v) \in A(G)$,
$t(e)$: the *tail vertex* of $e$, $(t(e)=u)$,
$h(e)$: the *head vertex* of $e$, $(h(e)=v)$, hence $e=(t(e), h(e))$.
For an arc set $A'$ $(\subseteq A(G))$,
$t(A')=\{t(e) \mid e \in A'\}$: the tail vertex set of arcs in $A'$,
$h(A')=\{h(e) \mid e \in A'\}$: the head vertex set of arcs in $A'$.
For a vertex $v$ in $V(G)$,
$\mathrm{IA}_G(v)=\{e \mid h(e)=v\}$: the incoming-arc set to $v$,
$\mathrm{OA}_G(v)=\{e \mid t(e)=v\}$: the outgoing-arc set from $v$,
$\mathrm{Fin}_G(v)=\{u \mid (u, v) \in A(G)\}=t(\mathrm{IA}_G(v))$: the tail vertex set of the incoming-arcs to $v$,
$\mathrm{Fout}_G(v)=\{w \mid (v, w) \in A(G)\}=h(\mathrm{OA}_G(v))$: the head vertex set of the outgoing-arcs from $v$,
$\mathrm{id}_G(v)=|\mathrm{Fin}_G(v)|$: the *indegree* of $v$,
$\mathrm{od}_G(v)=|\mathrm{Fout}_G(v)|$: the *outdegree* of $v$.

Throughout this paper, the subscript $G$ is occasionally omitted in the following notations if it is clear from context:

$\mathrm{IA}_G(v)$, $\mathrm{OA}_G(v)$, $\mathrm{Fin}_G(v)$, $\mathrm{Fout}_G(v)$, $\mathrm{id}_G(v)$, $\mathrm{od}_G(v)$, $P_G(u, v)$, $\mathrm{depth}_G(v)$, $n_G(v)$ and $C_G(u, v)$

($P_G(u, v)$, $\mathrm{depth}_G(v)$, $n_G(v)$ and $C_G(u, v)$ are defined below).

**Definition 2.** Two arcs $e_1$ and $e_2$ are said to be *independent* if $t(e_1) \neq t(e_2)$ and $h(e_1) \neq h(e_2)$.

For an arc set $A'$ ($\subseteq A(G)$), $A'$ is said to be *independent* if any two arcs in $A'$ are independent.

A $v_1$-$v_m$ *path* (or simply, a *path*) in $G$ is a sequence of distinct vertices $\langle v_1, v_2, \ldots, v_m \rangle$ such that: $(v_i, v_{i+1}) \in A(G)$ for each $i$ ($1 \leq i < m$).

The vertices $v_1$ and $v_m$ of a $v_1$-$v_m$ path are called the *start vertex* and the *end vertex*, respectively.

The *length* of the path $\langle v_1, \ldots, v_m \rangle$ is $m-1$, which is the number of arcs $(v_i, v_{i+1})$. Thus, a $v$-$v$ path $\langle v \rangle$ will be of length 0 by definition.

Two $u$-$v$ paths are said to be *internally-disjoint* if these paths do not share any vertex except for $u$ and $v$. For distinct vertices $u$ and $v$ in $G$, $P_G(u, v)$ denotes the maximum number of mutually internally-disjoint $u$-$v$ paths in $G$.

For vertex sets $V_1$ ($\subseteq V(G)$) and $V_2$ ($\subseteq V(G)$) of the same cardinality $m = |V_1| = |V_2|$, and a vertex $v$ ($\in V(G)$),

A $(V_1, V_2)$-*link* is a set of $m$ paths such that:

(i) any two distinct paths in the $(V_1, V_2)$-link do not share any vertex, including the start vertices and the end vertices, and

(ii) any path in the $(V_1, V_2)$-link is a $v_1$-$v_2$ path for some vertices $v_1 \in V_1$ and $v_2 \in V_2$. Note that the length of this path is 0 in the case of $v_1 = v_2$.

A $(v, V_1)$-*fanout* is a set of $m$ paths such that:

(i) any two distinct paths in the $(v, V_1)$-fanout do not share any vertex except for $v$, and

(ii) any path in the $(v, V_1)$-fanout is a $v - v_1$ path for some vertex $v_1$ in $V_1$. Note that the length of this path is 0 in the case of $v = v_1$.

A $(V_1, v)$-*fanin* is a set of $m$ paths such that:

(i) any two distinct paths in the $(V_1, v)$-fanin do not share any vertex except for $v$, and

(ii) any path in the $(V_1, v)$-fanin is a $v_1$-$v$ path for some vertex $v_1$ in $V_1$. Note that the length of this path is 0 in the case of $v = v_1$.

A digraph $G' = (V', A')$ is called a *subgraph* of $G = (V, A)$ if $V \supseteq V'$ and $A \supseteq A'$ hold. Especially, $G'$ is called a *spanning subgraph* of $G$ if $V' = V$ holds.

**Example 1.** Fig. 1 shows a $(V_1, V_2)$-link, a $(v, V_1)$-fanout and a $(V_1, v)$-fanin in $G$.

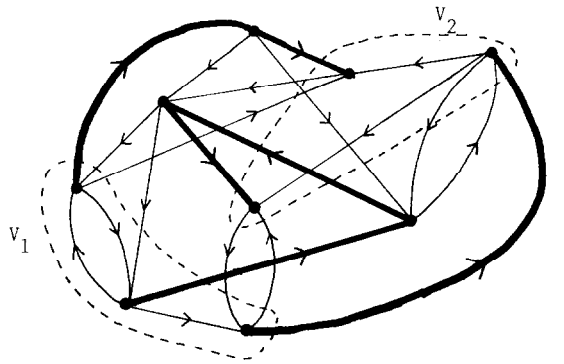
**Notation 3.** For $G = (V, A)$, a vertex set $V'$ ($\subseteq V$) and an arc set $A'$ ($\subseteq A$),

$G[V'] = (V', \{e \in V' \times V' \mid e \in A\})$: the *induced subgraph* of $G$ with the vertex set $V'$,

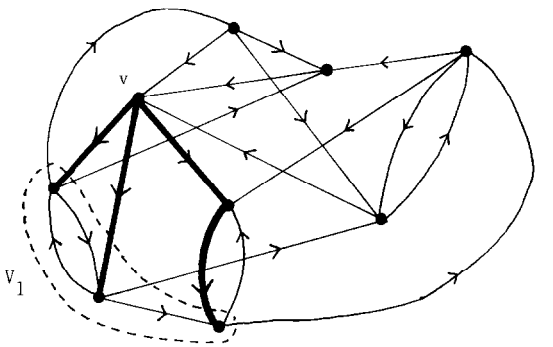
$G - V'$: the digraph $G[V - V']$,

$G - A'$: the digraph $(V, A - A')$,

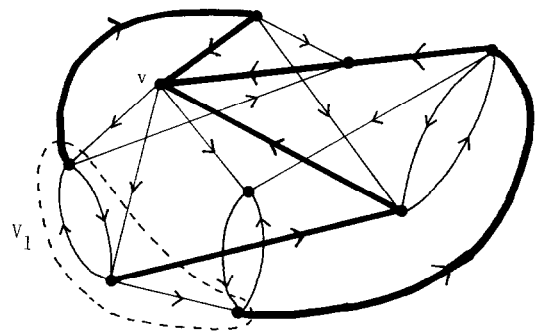
a $u$-$v[V']$ path: a $u$-$v$ path in $G[V']$, that is, a $u$-$v$ path $\langle u(=u_1), u_2, \ldots, v(=u_m) \rangle$ such that $u_i \in V'$ for each $i$ ($1 \leq i \leq m$),




(a) A ($V_1$,$V_2$)-link.




(b) A (v,$V_1$)-fanout.




(c) A ($V_1$,v)-fanin.

Fig. 1. A $(V_1, V_2)$-link, a $(v, V_1)$-fanout and a $(V_1, v)$-fanin.

a $(V_1, V_2)[V']$-link: a $(V_1, V_2)$-link in $G[V']$,
a $(v, V_1)[V']$-fanout: a $(v, V_1)$-fanout in $G[V']$,
a $(V_1, v)[V']$-fanin: a $(V_1, v)$-fanin in $G[V']$.

**Definition 3.** Two digraphs $G_1$ and $G_2$ are said to be *independent* if $V(G_1) \cap V(G_2) = \emptyset$ holds.

A digraph $G=(V,A)$ is said to be a *complete digraph*, denoted by $K_{|V|}$, if for any two distinct vertices $u$ and $v$, both $(u,v)$ and $(v,u)$ belong to $A$.

A *rooted directed tree* (or simply a *tree*) is a digraph $G=(V,A)$ such that

(i) there exists the root $v_0$ with $\mathrm{id}(v_0)=0$,

(ii) $\mathrm{id}(v)=1$ for any vertex $v$ other than $v_0$,

(iii) there exists a $v_0$–$v$ path for every $v \in V(G)$.

A tree $G$ is called an *m-ary tree* if $\mathrm{od}(v) \leq m$ for every $v \in V(G)$.

If $(u,v)$ is an arc of a tree, then $u$ is called the *father* of $v$ and $v$ is called a *son* of $u$. A vertex $v$ of a tree is called a *leaf* if $\mathrm{od}(v)=0$; otherwise it is called an *internal vertex*.

The *depth* of a vertex $v$ in a tree with the root $v_0$ is the length of the $v_0$–$v$ path.

The *subtree* of $G$ with a root $v$ is an induced subgraph of $G$ with a vertex set $\{u \mid \text{there exists a } v\text{–}u \text{ path in } G\}$.

**Notation 4.** $G=(V,A,v_0)$: (sometimes,) a tree $G=(V,A)$ with the root $v_0$.

$\mathrm{depth}_G(v)$: the depth of a vertex $v$ in a tree $G$.

$G(v)=(V(v),A(v))$: the subtree of $G$ with the root $v$ ($\in V(G)$).

$n_G(v)=|V(v)|$: the number of vertices in the subtree with the root $v$ in $G$; hence, $n_G(v)=|V(G(v))|$.

**Definition 4.** A vertex $u$ of a tree $G=(V,A)$ is called an *m-weak-centroid-vertex* if

(i) $n(u) \geq |V|-m$ and

(ii) $n(v) \leq m$ for every son $v$ of $u$.

An arc $(u,v)$ in a tree $G=(V,A)$ is called an *m-bridge* if $m \leq n(v) \leq |V|-m$.

**Example 2.** Fig. 2 shows examples of a 5-weak-centroid-vertex and a 5-bridge.

**Definition 5.** A tree is called an *ordered tree* if for every internal vertex $v$, there is a defined order of its sons.

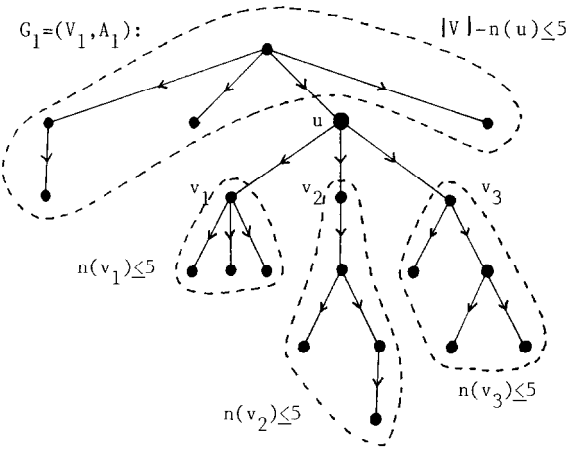
The *j-th son* of a vertex $v$ is denoted by $v^j$ $(1 \leq j \leq \mathrm{od}(v))$.

A path $\langle u_1, u_2, \ldots, u_m \rangle$ in an ordered tree $G$ is called the *rightmost path* of $G$ if $u_1$ is the root of $G$, $u_m$ is a leaf of $G$ and $u_{i+1}=u_i^{\mathrm{od}(u_i)}$ for each $i$ $(1 \leq i \leq m-1)$.

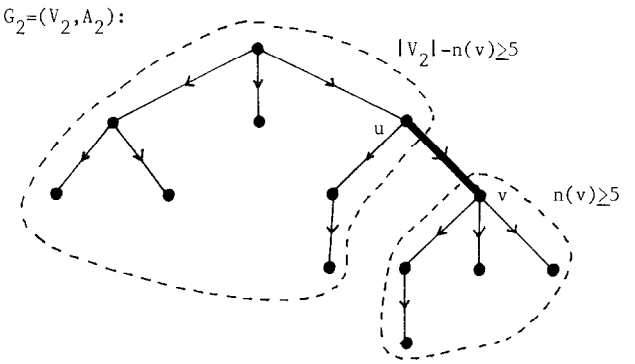
If an ordered tree $G$ satisfies $n(v^j) \leq n(v^{j+1})$ for each vertex $v$ and each $j$ $(1 \leq j \leq \mathrm{od}(v)-1)$, then the ordered tree is said to be in the *normal-ordering*.

**Example 3.** Fig. 3 shows a tree in the normal-ordering and its rightmost path.

**Notation 5.** For $G$, $\overline{m}$ denotes $m \bmod |V(G)|$.




(a) u is a 5-weak-centroid-vertex in $G_1$.




(b) (u,v) is a 5-bridge in $G_2$.

Fig. 2. A 5-weak-centroid-vertex and a 5-bridge.

**Definition 6.** For $G=(V,A)$, consider a bijection $f$ from $V$ to a set of integers $[0\cdots|V|-1]$. If $f$ satisfies $0<\overrightarrow{f(v)-f(u)}\leq m$ for each arc $(u,v)$, then $f$ is called an *m-bandwidth-order*. Especially, if $f$ satisfies $0<f(v)-f(u)\leq m$ for each arc $(u,v)$, then $f$ is called an *m-strong-bandwidth-order*.

The *depth-first-order* of an ordered tree $G=(V,A,v_0)$ is the bijection $f$ from $V$ to $[0\cdots|V|-1]$ such that

(i) $f(v_0)=0$, and

(ii) $f(v)=\sum_{j=1}^{i-1} n(v_0^j)+f'(v)+1 \quad (v\in V(v_0^i))$

where $f'$ is the depth-first-order of $G(v_0^i)$.

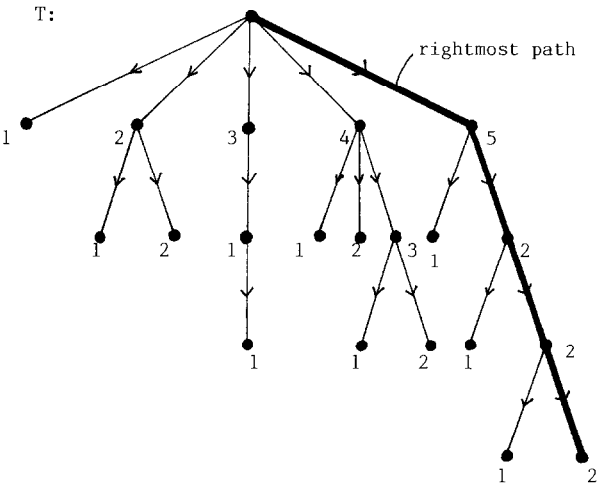



Fig. 3. A tree $T$ in the normal-ordering and its rightmost path.

Namely, the depth-first-order numbers vertices from 0 to $|V|-1$ in the order of being visited by depth-first search in [1].

The *breadth-first-order* of an ordered tree $G=(V,A,v_0)$ is the bijection $f$ from $V$ to $[0\cdots|V|-1]$ such that

$$f(v)=|\{w\,|\,\mathrm{depth}(w)<\mathrm{depth}(v)\}|$$
$$+|\{w\,|\,\mathrm{depth}(w)=\mathrm{depth}(v) \text{ and } f(w')<f(v') \text{ for the parent } w' \text{ of } w \text{ and the parent } v' \text{ of } v\}|$$
$$+|\{w\,|\,\mathrm{depth}(w)=\mathrm{depth}(v) \text{ and } w \text{ and } v \text{ have the same parent } u \text{ and } w=u^i \text{ and } v=u^j \text{ for some } i<j\}|$$

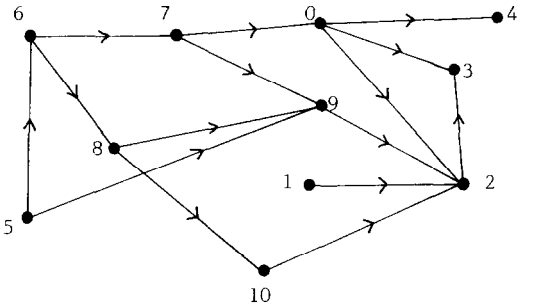
By the breadth-first-order, vertices are numbered from 0 to $|V|-1$ in the order of being visited by breadth-first search in [1].

A digraph $G$ is called to be *m-regular* if $\mathrm{od}(v)=\mathrm{id}(v)=m$ holds for each vertex $v$ in $V(G)$.

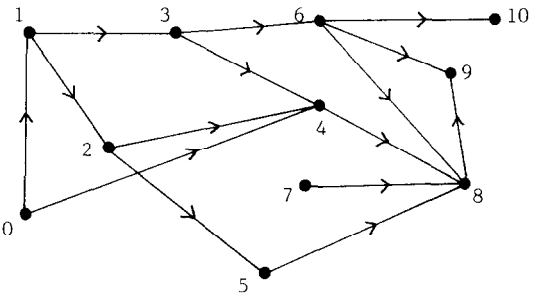
A digraph $G$ is called to be *m-vertex-connected* (or simply *m-connected*) if there exist at least $m$ internally-disjoint $u$–$v$ paths for any distinct vertices $u$ and $v$ in $V(G)$.

**Example 4.** Fig. 4 shows examples of a 4-bandwidth-order and a 4-strong-bandwidth-order of $G$.

**Example 5.** In Fig. 5, examples of the depth-first-order and the breadth-first-order of an ordered tree $G$ are shown.




(a) A 4-bandwidth-order.




(b) A 4-strong-bandwidth-order.

Fig. 4. A 4-bandwidth-order and a 4-strong-bandwidth-order.

The following propositions hold.

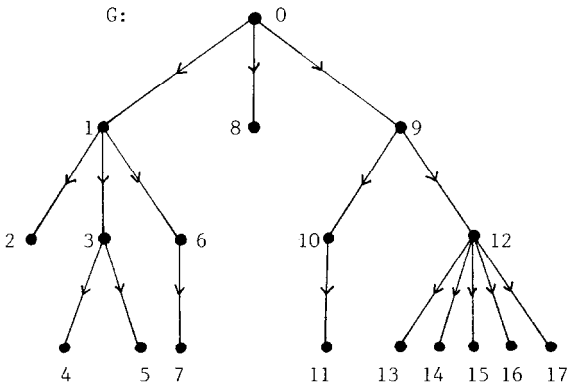
**Proposition 1** [4]. *If $G=(V,A)$ is not a complete digraph, then*

$$\min_{\substack{(u,v)\in V\times V-A \\ u\neq v}} P(u,v) = \min_{\substack{u,v\in V \\ u\neq v}} P(u,v).$$

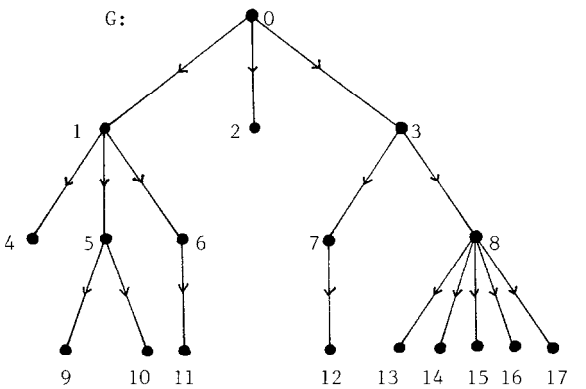
Namely, Proposition 1 means that the smallest value of $P(u,v)$ for two distinct vertices $u$ and $v$ occurs also for some two distinct vertices $u$ and $v$ such that $(u,v)\notin A$.

**Definition 7.** For $G=(V,A)$ and two distinct vertices $u$ and $v$ such that $(u,v)\notin A$, $C_G(u,v)$ denotes a vertex set with minimum cardinality to *separate $v$ from $u$*, that is, there exists no $u$-$v$ path in $G-C_G(u,v)$.

**Proposition 2** [4]. *In $G=(V,A)$, if $(u,v)\notin A$, then $|C_G(u,v)|=P_G(u,v)$.*




(a) The depth-first-order of G.




(b) The breadth-first-order of G.

Fig. 5. The depth-first-order and the breadth-first-order.

Proposition 2 means that for any two distinct vertices $u$ and $v$ such that $(u, v) \notin A(G)$, the minimum number of vertices to separate $v$ from $u$ is equal to the maximum number of internally disjoint $u-v$ paths.

Proposition 1 and 2 mean that it suffices for proving that $G=(V, A)$ which is not a complete digraph is $m$-connected to show that $P(u, v) \geq m$ or $|C(u, v)| \geq m$ holds for any two distinct vertices $u$ and $v$ such that $(u, v) \notin A$.

By the similar discussion to that for an undirected graph in [2], the following propositions can be proved.

**Proposition 3.** *A digraph $G=(V, A)$ is $m$-connected if and only if $|V| \geq m+1$ and there exists a $(v, V')$-fanout for any vertex $v$ and any vertex set $V'$ $(\subseteq V)$ such that $|V'|=m$.*

**Proposition 4.** *A digraph $G=(V,A)$ is m-connected if and only if $|V|\geq m+1$ and there exists a $(V',v)$-fanin for any vertex $v$ and any vertex set $V'$ ($\subseteq V$) such that $|V'|=m$.*

## 3. The number of arcs to be added

**Theorem 1.** *If a k-connected digraph is constructed from a digraph $G=(V,A)$ by adding arcs, then the set of arcs $A'$ to be added satisfies*

$$|A'|\geq \max\left(\sum_{v\in V}\max(k-\mathrm{id}_G(v),0),\ \sum_{v\in V}\max(k-\mathrm{od}_G(v),0)\right).$$

**Proof.** Theorem 1 is immediately proved from the fact that each vertex $v$ in a $k$-connected digraph $G'$ constructed from $G$ satisfies

$$\mathrm{id}_{G'}(v)\geq\max(\mathrm{id}_G(v),k)\quad\text{and}\quad\mathrm{od}_{G'}(v)\geq\max(\mathrm{od}_G(v),k).\qquad\square$$

From Theorem 1, we can get the following corollary.

**Corollary 1.** *Let $G'=(V,A\cup A')$ $(A\cap A'=\emptyset)$ be an augmented digraph of $G=(V,A)$. If $G'$ is k-regular and k-connected, then the set of arcs $A'$ is one of the solutions of* $k$-VCUAP *for $G$.*

Corollary 1 means that constructing a $k$-regular and $k$-connected digraph from a given digraph $G$ by adding arcs is one of the ways to solve $k$-VCUAP for $G$. In the following two sections, $k$-regular and $k$-connected digraphs, which are of extreme importance to our discussion, are introduced.

## 4. Super daisy chain

In this section, a $k$-regular and $k$-connected digraph called a super daisy chain is defined and some of its properties are discussed.

**Definition 8.** For integers $n$ and $k$ $(n>k\geq 1)$, a *super daisy chain* $D_{n,k}=(V,A)$ is defined as follows:

$$V=\{0,1,\ldots,n-1\},$$

$$A=\bigcup_{i=1}^{k}A_i(D_{n,k})\quad\text{where } A_i(D_{n,k})=\{(j,\overline{j+i})\mid 0\leq j\leq n-1\}.$$

The sequence of vertices $\langle u,\overline{u+1},\overline{u+2},\ldots,\overline{u+(m-1)}\rangle$ is called an *m-consecutive vertex sequence*. $\mathrm{Int}(u,v)$ stands for the set of vertices in the $m$-consecutive vertex sequence $\langle u,\overline{u+1},\overline{u+2},\ldots,v\rangle$ with $\overline{u+(m-1)}=v$.

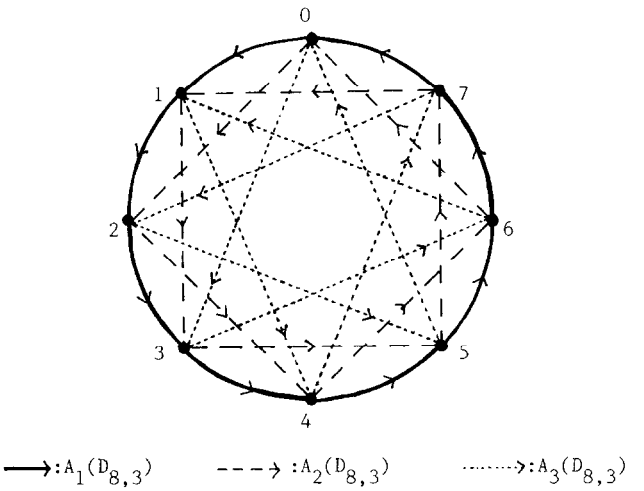



Fig. 6. A super daisy chain $D_{8,3}$.

**Example 6.** Fig. 6 shows $D_{8,3}$. In $D_{8,3}$, $\langle 6, 7, 0, 1, 2\rangle$ is a 5-consecutive sequence and $\mathrm{Int}(6, 2) = \{6, 7, 0, 1, 2\}$.

**Lemma 1.** *In $D_{n,k}$ $(n > k \geq 1)$, let $V_1$ and $V_2$ be arbitrary vertex sets such that $|V_1| = |V_2| \leq k$. There exists a $(V_1, V_2)$-link in $D_{n,k}$.*

**Proof.** Let $|V_1| = |V_2| = p$ $(1 \leq p \leq k)$, and let

$$s_i = |V_1 \cap \mathrm{Int}(j, \overline{j+i})|, \qquad t_i = |V_2 \cap \mathrm{Int}(j, \overline{j+i})| \quad \text{and} \quad d_i = s_i - t_i$$

for $j \in V_1$ and each $i$ $(0 \leq i < n)$.

We can choose a $j$ such that $d_i = s_i - t_i \geq 0$ for all $i$ $(0 \leq i < n)$ and we can assume $j = 0$ by renumbering (without loss of generality).

Let $q$ $(p - 1 \leq q \leq n - 1)$ denote the minimum among $i$ such that $t_i = p$. By mathematical induction on $p$, we can prove the lemma in a stronger form: "There exists a $(V_1, V_2)[\mathrm{Int}(0, q)]$-link".

(Basis) $p = 1$: In this case, $V_1 = \{0\}$ and $V_2 = \{q\}$. There exists the $(V_1, V_2)[\mathrm{Int}(0, q)]$-link $\{\langle 0, 1, \ldots, q\rangle\}$.

(Inductive hypothesis) $p = m$ $(1 \leq m < k)$: There exists a $(V_1, V_2)[\mathrm{Int}(0, q)]$-link.

(Inductive step) $p = m + 1$: The property is shown by mathematical induction on $q$.

(Basis) $q = p - 1$: In this case, $V_1 = V_2 = \{0, 1, \ldots, p - 1\}$ and there exists the $(V_1, V_2)[\mathrm{Int}(0, q)]$-link $\{\langle 0\rangle, \langle 1\rangle, \ldots, \langle p - 1\rangle\}$.

(Inductive hypothesis) $p - 1 \leq q \leq h$ $(p - 1 \leq h < n - 1)$: There exists a $(V_1, V_2)[\mathrm{Int}(0, q)]$-link.

(Inductive step) $q = h + 1$:

(Case 1) $q \in V_1$: Letting $V_1' = V_1 - \{q\}$ and $V_2' = V_2 - \{q\}$, it suffices to show that there exists a $(V_1', V_2')$[Int$(0, q-1)$]-link. Here, $|V_1'| = |V_2'| = p - 1 = m$ holds.

For each $i$ $(0 \le i \le n-1)$, let

$$s_i' = |V_1' \cap \mathrm{Int}(0, i)|, \qquad t_i' = |V_2' \cap \mathrm{Int}(0, i)| \quad \text{and} \quad d_i' = s_i' - t_i'.$$

It is evident that $t_h' = t_h = p - 1$ holds.

Letting $q'$ be the minimum $i$ such that $t_i' = p - 1$, then $q' \le h$. From

$$d_i' = s_i' - t_i' = s_i - t_i \ge 0 \quad (0 \le i \le h) \quad \text{and}$$

$$d_i' = s_i' - t_i' = (s_i - 1) - (t_i - 1) = d_i \ge 0 \quad (h+1 \le i \le n-1),$$

it follows that $d_i' \ge 0$ for each $i$ $(0 \le i \le n-1)$. From the inductive hypothesis for $p = m$, there exists a $(V_1', V_2')$[Int$(0, q-1)$]-link. Therefore, there exists a $(V_1, V_2)$[Int$(0, q)$]-link.

(Case 2) $q \notin V_1$: Let the maximum $i$ satisfying $i \in V - V_2$ and $0 \le i \le h$ be $r$. It follows from $|V_2| = p$ that $h + 1 - p \le r$. For each $i$ $(r < i \le h+1)$, it follows from $i \in V_2$ that $d_i \ge 1$ and $d_r \ge 1$. Letting $V_2' = (V_2 - \{h+1\}) \cup \{r\}$, then $|V_2'| = p$.

For each $i$ $(0 \le i \le n-1)$, let

$$t_i' = |V_2' \cap \mathrm{Int}(0, i)|, \quad \text{and} \quad d_i' = s_i - t_i'.$$

Here, $t_h' = t_h + 1 = p$ holds.

Letting the minimum $i$ such that $t_i' = p$ be $q'$, then $q' \le h$. From

$$d_i' = s_i - t_i' = s_i - t_i = d_i \ge 0 \quad (0 \le i \le r-1),$$

$$d_i' = s_i - t_i' = s_i - (t_i + 1) = d_i - 1 \ge 0 \quad (r \le i \le h) \quad \text{and}$$

$$d_i' = s_i - t_i' = s_i - t_i = d_i \ge 0 \quad (h+1 \le i \le n-1),$$

it follows that $d_i' \ge 0$ holds for each $i$ $(0 \le i \le n-1)$. By the inductive hypothesis for $p - 1 \le q \le h$, there exists a $(V_1, V_2')$[Int$(0, q-1)$]-link. It follows from $1 \le h + 1 - r \le p \le k$ that $(r, h+1) \in A_{h+1-r}(D_{n,k})$, and there exists a $(V_1, V_2)$[Int$(0, q)$]-link. □

**Theorem 2.** *The super daisy chain $D_{n,k}$ $(n > k)$ is $k$-regular and $k$-connected.*

**Proof.** By Definition 8, $D_{n,k}$ is obviously $k$-regular.

(Case 1) $n = k + 1$: Then $D_{n,k} = K_n$. Therefore, $D_{n,k}$ is clearly $k$-connected.

(Case 2) $n > k + 1$: Let $u$ and $v$ be any two distinct vertices of $D_{n,k}$ such that $(u, v) \notin A(D_{n,k})$. Letting $U = \mathrm{Fout}(u)$ and $V = \mathrm{Fin}(v)$, then $|U| = |V| = k$. By Lemma 1, there exists a $(U, V)$-link. This fact means that there exist $k$ internally-disjoint $u$–$v$ paths in $D_{n,k}$. Therefore, $D_{n,k}$ is $k$-connected. □

It follows from Corollary 1 and Theorem 2 that $k$-VCUAP for $G$ is solved if $D_{|V(G)|,k}$ is constructed from $G$ by adding arcs. However, it is easy to show that there exists the $k$-ary tree from which $D_{|V|,k}$ cannot be constructed by adding arcs. For example, the ternary tree in Fig. 7 cannot be augmented to $D_{10,3}$.

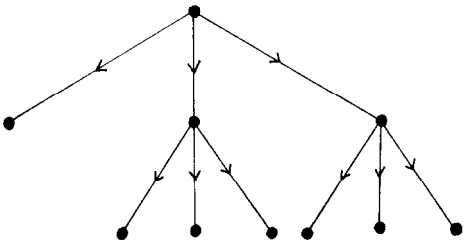


Fig. 7. A ternary tree from which $D_{10,3}$ cannot be constructed by adding arcs.

Moreover, we can achieve the following theorem.

**Theorem 3.** *The decision problem for a given k and a given $G=(V,A)$ to decide whether or not $D_{|V|,k}$ can be constructed from G by adding arcs is NP-complete. Even if G* is restricted to a binary tree, it remains NP-complete.

In advance of proving Theorem 3, we will show two lemmas.

**Lemma 2.** *$G=(V,A)$ ($|V|>k$) can be augmented to $D_{|V|,k}$ by adding arcs if and only if there exists a k-bandwidth-order $f: V\to[0\cdots|V|-1]$.*

Lemma 2 obviously holds.

**Lemma 3.** *For a tree $T=(V,A,v_0)$, there exists a k-strong-bandwidth-order $f: V\to[0\cdots|V|-1]$ if and only if there exists a k-bandwidth-order $g: V\to[0\cdots|V|-1]$.*

**Proof.** (Only if). It is clear that there exists a $k$-strong-bandwidth-order only if there exists a $k$-bandwidth-order.

(If). Let $g': V\to[0\cdots|V|-1]$ be defined as $g'(v)=\overline{g(v)-g(v_0)}$. Since $g$ is a bijection, $g'$ is also a bijection. From $\overline{g'(v)-g'(u)}=\overline{g(v)-g(u)}$ for any two distinct vertices $u$ and $v$, $g'$ is a $k$-bandwidth-order. Define sets of arcs $A_1$ and $A_2$ as follows:

$$A_1=\{(u,v)\in A\mid g'(v)>g'(u)\} \quad \text{and}$$

$$A_2=A-A_1=\{(u,v)\in A\mid g'(v)\le g'(u)\}.$$

For each $i$ ($0\le i\le|V|-1$), let $v_i$ stand for the vertex $v$ such that $g'(v)=i$. As $T$ is a tree, there exists the unique $v_0$-$v_i$ path for each $v_i$. Let $m_i$ be the number of arcs of $A_2$ which appear on the $v_0$-$v_i$ path.

To define $f: V\to[0\cdots|V|-1]$, we will introduce a linear ordering $\prec$ defined on $V$ as follows:

$$v_i\prec v_j \quad \text{if } m_i<m_j \quad \text{or} \quad \text{if } m_i=m_j \text{ and } g'(v_i)<g'(v_j).$$

In other words, this ordering relation is the lexicographic order with a major key $m_i$ and a minor key $g'(v_i)$. Consider the vertex sequence of $V$ where all vertices of $V$ are sorted using the ordering relation $\prec$ in increasing order, and define $f$ as follows:

$$f(v_i)=i' \quad \text{if } v_i \text{ is the } (i'+1)\text{-th vertex in the sequence.}$$

Therefore, it is clear that $f$ is a bijection. The bijection $f$ is also described by the following equation:

$$\begin{aligned}
f(v_i) &= |\{j \mid 0\le j\le |V|-1 \text{ and } m_j<m_i\}| \\
&\quad + |\{j \mid 0\le j\le |V|-1,\ m_j=m_i \text{ and } g'(v_j)<g'(v_i)\}| \\
&= |\{j \mid 0\le j\le |V|-1 \text{ and } m_j<m_i\}| + |\{j \mid 0\le j\le i-1 \text{ and } m_j=m_i\}| \\
&\quad \text{(Recall that } v_j \text{ is the vertex such that } g'(v_j)=j\ (0\le j\le |V|-1).) \\
&= |\{j \mid 0\le j\le i-1 \text{ and } m_j\le m_i\}| \\
&\quad + |\{j \mid i+1\le j\le |V|-1 \text{ and } m_j<m_i\}| \\
&= i-|\{j \mid 0\le j\le i-1 \text{ and } m_j>m_i\}| \\
&\quad + |\{j \mid i+1\le j\le |V|-1 \text{ and } m_j<m_i\}|.
\end{aligned}$$

In the following, it is shown that $f$ is a $k$-strong-bandwidth-order, that is, $0<f(v_q)-f(v_p)\le k$ holds for any arc $(v_p, v_q)$ in $A$.

(Case 1) $(v_p, v_q)\in A_1$: $0<q-p\le k$ and $m_p=m_q$ hold. Then, $f(v_q)-f(v_p)=q-p-|\{j \mid p+1\le j\le q-1 \text{ and } m_j\ne m_p\}|$. It follows from $0\le|\{j \mid p+1\le j\le q-1 \text{ and } m_j\ne m_p\}|\le q-p-1$ that $0<f(v_q)-f(v_p)\le k$ holds.

(Case 2) $(v_p, v_q)\in A_2$: Here $p>q$, $0<\overline{q-p}\le k$ and $m_q=m_p+1$ hold. Then,

$$\begin{aligned}
f(v_q)-f(v_p) = q-p &+ |\{j \mid 0\le j\le q \text{ and } m_j=m_q\}| \\
&+ |\{j \mid q+1\le j\le p-1 \text{ and } m_j>m_p\}| \\
&+ |\{j \mid q+1\le j\le p-1 \text{ and } m_j<m_q\}| \\
&+ |\{j \mid p\le j\le |V|-1 \text{ and } m_j=m_p\}|.
\end{aligned}$$

It follows from

$$\begin{aligned}
&1\le|\{j \mid 0\le j\le q \text{ and } m_j=m_q\}|\le q+1, \\
&1\le|\{j \mid p\le j\le |V|-1 \text{ and } m_j=m_p\}|\le |V|-p \quad \text{and} \\
&|\{j \mid q+1\le j\le p-1 \text{ and } m_j>m_p\}|+|\{j \mid q+1\le j\le p-1 \text{ and } m_j<m_q\}| \\
&\quad = p-q-1
\end{aligned}$$

that $0<f(v_q)-f(v_p)\le k$ holds. □

**Proof of Theorem 3.** The decision problem for a given $k$ and a given $G=(V,A)$ to decide whether or not there exists a $k$-strong-bandwidth-order for $G$ is called a directed bandwidth problem. It is shown in [6] that the directed bandwidth problem is NP-complete even for a binary tree. From Lemma 2 and Lemma 3, Theorem 3 is proved. □

In the rest of this section, we consider $D_{n,k}$'s with some vertices and/or arcs deleted and present some of its properties.

**Definition 9.** Let $D'$ denote the digraph $(D_{n,k}-V')-A'$ where $V'\subseteq V(D_{n,k})$ and $A'\subseteq A(D_{n,k})$. The sequence $\langle v_1, v_2, \ldots, v_m\rangle$ of vertices in $V(D')$ is called an *m-pseudo-consecutive vertex sequence* of $D'$, if $\mathrm{Int}(v_i, v_{i+1})-\{v_i, v_{i+1}\}\subseteq V'$ holds for each $i$ $(1\leq i\leq m-1)$.

**Example 7.** Fig. 8 shows a 4-pseudo-consecutive vertex sequence $\langle 2,4,6,7\rangle$ of $D_{12,3}-\{3,5,9,10\}$.

**Lemma 4.** *For $D_{n,k}=(V,A)$ $(n>k\geq 2)$, let $V'$ be any subset of $V$ such that $|V'|=k-2$ and let $A'$ be any independent subset of $A$. Let $D'=(D_{n,k}-V')-A'$. Let $v_0$ be any vertex in $D'$ and let $\langle v_1, v_2\rangle$ be any 2-pseudo-consecutive vertex sequence in $D'$. Then,*

(a) *There exists either a $v_0$-$v_1$ $[\mathrm{Int}(v_0, v_1)]$ path or a $v_0$-$v_2$ $[\mathrm{Int}(v_0, v_2)]$ path in $D'$, that is, there exists either a $v_0$-$v_1$ path only going through the vertices in $\mathrm{Int}(v_0, v_1)$ or a $v_0$-$v_2$ path only going through the vertices in $\mathrm{Int}(v_0, v_2)$.*

(b) *There exists either a $v_1$-$v_0$ $[\mathrm{Int}(v_1, v_0)]$ path or a $v_2$-$v_0$ $[\mathrm{Int}(v_2, v_0)]$ path in $D'$.*

**Proof.** We will prove only part (a) here. Part (b) is similarly proved.

Without loss of generality, we can assume that $\overline{v_1-v_0}<\overline{v_2-v_0}$ holds. It is proved

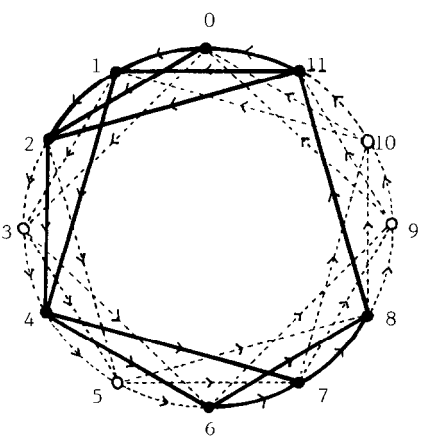



Fig. 8. A 4-pseudo-consecutive vertex sequence $\langle 2,4,6,7\rangle$ in $D_{12,3}-\{3,5,9,10\}$.

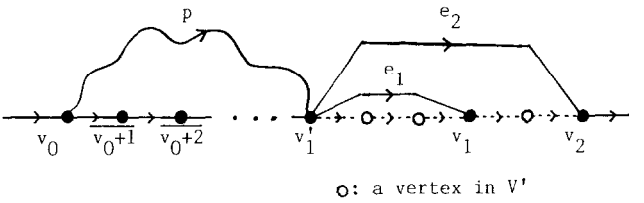



Fig. 9. A $v_0$-$v_1'$ path $p$ and arcs $e_1$ and $e_2$.

by using mathematical induction on $q$ where $q = |\mathrm{Int}(v_0, v_2) - V'|$.

(Basis) $q = 2$: Then, $v_0 = v_1$, and part (a) holds obviously.

(Inductive hypothesis) $q = m$ $(m \geq 2)$: Assume part (a) holds.

(Inductive step) $q = m + 1$: Let $\langle v_1', v_1 \rangle$ be a 2-pseudo-consecutive vertex sequence in $D'$. Then, $|\mathrm{Int}(v_0, v_1) - V'| = m$. From the inductive hypothesis, there exists either a $v_0$-$v_1$[$\mathrm{Int}(v_0, v_1)$] path or a $v_0$-$v_1'$[$\mathrm{Int}(v_0, v_1')$] path.

(Case 1) A $v_0$-$v_1$[$\mathrm{Int}(v_0, v_1)$] path exists: Part (a) holds obviously.

(Case 2) A $v_0$-$v_1'$[$\mathrm{Int}(v_0, v_1')$] path $p$ exists: From $|V'| = k-2$ and $\{v_1', v_1, v_2\} \subseteq V - V'$, both $e_1 = (v_1', v_1)$ and $e_2 = (v_1', v_2)$ belong to $A(D_{n,k} - V')$. It follows that both $e_1$ and $e_2$ do not belong to $A'$ because $A'$ is independent. If $e_1 \notin A'$, then a $v_0$-$v_1$[$\mathrm{Int}(v_0, v_1)$] path can be constructed from the path $p$ and the arc $e_1$. If $e_2 \notin A'$, then a $v_0$-$v_2$[$\mathrm{Int}(v_0, v_2)$] path can be constructed from the path $p$ and the arc $e_2$. (See Fig. 9.) □

**Lemma 5.** *For $D_{n,k} = (V, A)$ $(n - 3 \geq k \geq 3)$, let $A'$ be any independent subset of $A$. Then, $D_{n,k} - A'$ is $(k-1)$-connected.*

**Proof.** Let $D = D_{n,k} - A'$ and assume that $|C_D(v_1, v_2)| \leq k - 2$ for some distinct vertices $v_1$ and $v_2$ such that $(v_1, v_2) \notin A(D)$. In the following, we shall show that there exists a $v_1 - v_2$ path in $D' = D - C_D(v_1, v_2)$ under the assumption, which incurs a contradiction.

Let us consider two cases:

$$\text{case 1:} \quad |\mathrm{Fin}_{D_{n,k}}(v_2) \cap C_D(v_1, v_2)| \leq k - 3, \tag{1}$$

$$\text{and case 2:} \quad |\mathrm{Fin}_{D_{n,k}}(v_2) \cap C_D(v_1, v_2)| = k - 2. \tag{2}$$

(Case 1): Let $\langle w_1, w_2, w_3, v_2 \rangle$ be a 4-pseudo-consecutive vertex sequence in $D'$. From (1), $\overline{v_2 - w_i} \leq k$, that is, $e_i = (w_i, v_2) \in A$, for each $i$ $(1 \leq i \leq 3)$. As $A'$ is independent, at most one of the three arcs belongs to $A'$. (See Fig. 10(a).)

(Case 1.1) $e_3 \in A(D')$, that is, $e_3 \notin A'$: From Lemma 4, there exists either a $v_1$-$w_3$ path or a $v_1$-$v_2$ path in $D'$. Therefore, there exists a $v_1$-$v_2$ path, which is a contradiction.

(Case 1.2) $e_3 \notin A(D')$: Then, both $e_1$ and $e_2$ are arcs of $D'$. From Lemma 4, there exists either a $v_1$-$w_1$ path or a $v_1$-$w_2$ path in $D'$. Therefore, there exists a $v_1$-$v_2$

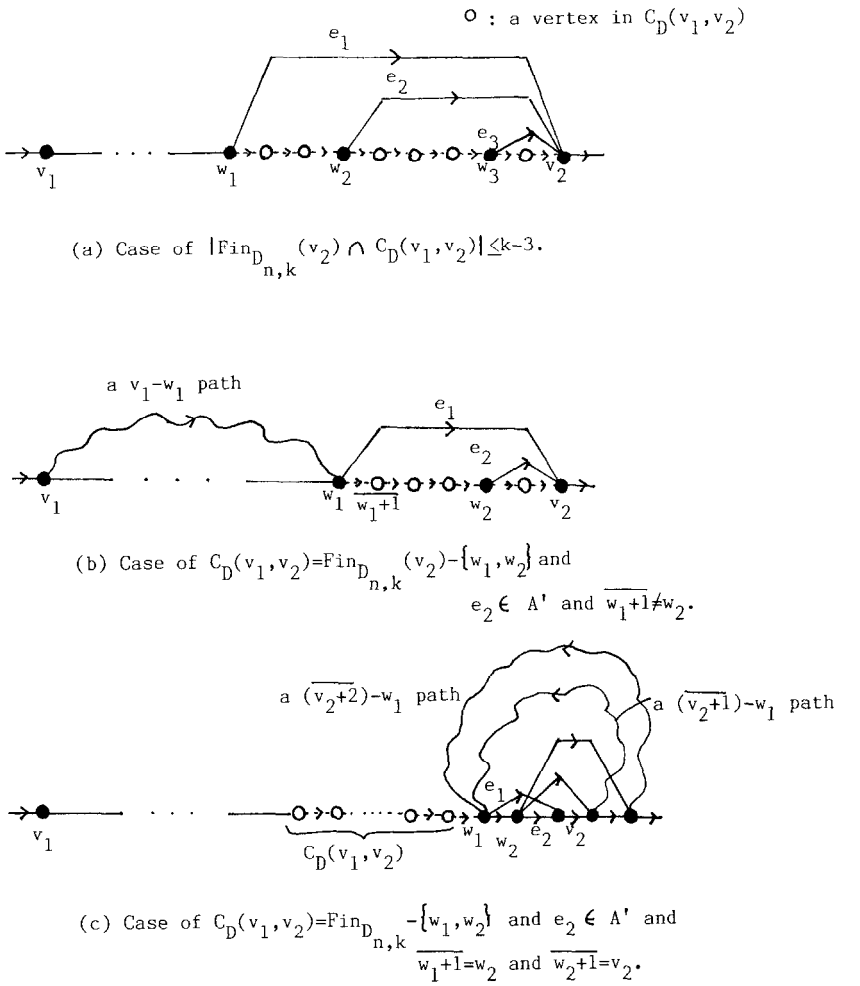



Fig. 10. Figures for the proof of Lemma 5.

path, which is a contradiction.

(Case 2): Let $\langle w_1, w_2, v_2\rangle$ be a 3-pseudo-consecutive vertex sequence in $D'$. From (2), $e_1=(w_1,v_2)\in A$, $e_2=(w_2,v_2)\in A$ and

$$C_D(v_1,v_2)=\mathrm{Fin}_{D_{n,k}}(v_2)-\{w_1,w_2\}. \tag{3}$$

As $A'$ is independent, at most one of $e_1$ and $e_2$ belongs to $A'$.

(Case 2.1) $e_2\in A(D')$: From Lemma 4, there exists either a $v_1$-$w_2$ path or a $v_1$-$v_2$ path in $D'$. Therefore, there exists a $v_1$-$v_2$ path, which is a contradiction.

(Case 2.2) $e_2\notin A(D')$, that is, $e_2\in A'$: Then, $e_1\in A(D')$.

(Case i) $\langle w_1, w_2, v_2\rangle$ is not a 3-consecutive vertex sequence in $D_{n,k}$: In the case of $\overline{w_1+1}\neq w_2$, $\overline{w_1+1}\in C_D(v_1,v_2)$ from (3). From $n\geq k+3$, $\overline{w_1+1}\notin \mathrm{Fin}_{D_{n,k}}(w_1)$. It

follows that $|\mathrm{Fin}_{D_{n,k}}(w_1) \cap C_D(v_1, v_2)| \leq k-3$, then there exists a $v_1$-$w_1$ path in $D'$ from Case 1. As $e_1 = (w_1, v_2)$ belongs to $A(D')$, there exists a $v_1$-$v_2$ path, which is a contradiction. (See Fig. 10(b).)

Similarly, it is shown that there exists a $v_1$-$v_2$ path in the case of $\overline{w_1+1} = w_2$ and $\overline{w_2+1} \neq v_2$.

(Case ii) $\langle w_1, w_2, v_2 \rangle$ is a 3-consecutive vertex sequence in $D_{n,k}$: Then, $w_2 = \overline{w_1+1}$, $v_2 = \overline{w_2+1}$ and $C_D(v_1, v_2) = \mathrm{Int}(\overline{v_2-k}, \overline{v_2-3})$ hold. It follows from $e_2 \notin A(D')$, $n-3 \geq k \geq 3$ and the fact that $A'$ is independent that $(w_2, \overline{v_2+1}) \in A(D')$ and $(w_2, \overline{v_2+2}) \in A(D')$. From Lemma 4, either a $(\overline{v_2+1})$-$w_1$ path or a $(\overline{v_2+2})$-$w_1$ path exists in $D'$. Therefore, there exists a $w_2$-$w_1$ path in $D'$. It follows from $e_1 \in A(D')$ that there exists a $w_2$-$v_2$ path. As there exists either a $v_1$-$w_2$ path or a $v_1$-$v_2$ path from Lemma 4, there exists a $v_1$-$v_2$ path in $D'$, which is a contradiction. (See Fig. 10(c).) □

## 5. Digraph operation and connectivity

This section introduces two operations to construct a $k$-regular and $k$-connected digraph from

(i) two $k$-regular and $k$-connected digraphs, or

(ii) $D_{n,k}$ and a vertex.

Also, a class of $k$-regular and $k$-connected digraphs called $k$-B-digraphs is defined as that of digraphs constructed by the above operations from $D_{n,k}$'s.

**Definition 10.** The digraph composition operation $\mathrm{dmergel}(G_1, \mathrm{IA}_1, \mathrm{OA}_1, G_2, \mathrm{OA}_2, \mathrm{IA}_2)$[1] is defined for independent digraphs $G_1 = (V_1, A_1)$ and $G_2 = (V_2, A_2)$ and for arc sequences $\mathrm{IA}_1 = \langle\langle \mathrm{IA}_{G_1}(v_1) \rangle\rangle$, $\mathrm{OA}_1 = \langle\langle \mathrm{OA}_{G_1}(v_1) \rangle\rangle$, $\mathrm{OA}_2 = \langle\langle \mathrm{OA}_{G_2}(v_2) \rangle\rangle$ and $\mathrm{IA}_2 = \langle\langle \mathrm{IA}_{G_2}(v_2) \rangle\rangle$ where $v_1 \in V_1$ and $v_2 \in V_2$ such that $\mathrm{id}_{G_1}(v_1) = \mathrm{od}_{G_2}(v_2) = j$ and $\mathrm{od}_{G_1}(v_1) = \mathrm{id}_{G_2}(v_2) = m$. It constructs $G = (V, A)$ as follows: (see Fig. 11)

$$V = V_1 \cup V_2 - \{v_1, v_2\},$$

$$A = ((A_1 \cup A_2) - (\mathrm{IA}_{G_1}(v_1) \cup \mathrm{OA}_{G_1}(v_1) \cup \mathrm{IA}_{G_2}(v_2) \cup \mathrm{OA}_{G_2}(v_2))$$
$$\cup \{(t((\mathrm{IA}_1)_i), h((\mathrm{OA}_2)_i)) \mid 1 \leq i \leq j\}$$
$$\cup \{(t((\mathrm{IA}_2)_i), h((\mathrm{OA}_1)_i)) \mid 1 \leq i \leq m\}.$$

**Definition 11.** For a digraph $G_1 = (V_1, A_1)$, a vertex $v \notin V_1$ and an independent arc set $A_1'$ ($\subseteq A_1$), the digraph composition operation $\mathrm{dmerge2}(G_1, A_1', v)$[2] is the one to construct the digraph $G = (V, A)$ as follows: (see Fig. 12)

[1] The operation dmergel and the operation merge2 in [10], [12] and [13] are the same.

[2] The operation dmerge2 and the operation merge3 in [10] are the same.

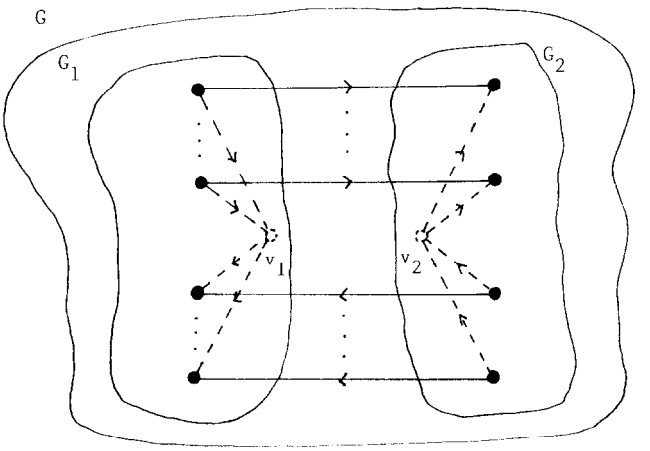



Fig. 11. A digraph composition operation dmerge1$(G_1, \mathrm{IA}_1, \mathrm{OA}_1, G_2, \mathrm{OA}_2, \mathrm{IA}_2)$.

$$V = V_1 \cup \{v\},$$
$$A = (A_1 - A_1') \cup \{(t(e), v) \mid e \in A_1'\} \cup \{(v, h(e)) \mid e \in A_1'\}.$$

The digraph obtained by the operation dmerge1$(G_1, \mathrm{IA}_1, \mathrm{OA}_1, G_2, \mathrm{OA}_2, \mathrm{IA}_2)$ is sometimes expressed by dmerge1$(G_1, \mathrm{IA}_1, \mathrm{OA}_1, G_2, \mathrm{OA}_2, \mathrm{IA}_2)$. The same for dmerge2.

Also, dmerge1$(G_1, \mathrm{IA}_1, \mathrm{OA}_1, G_2, \mathrm{OA}_2, \mathrm{IA}_2)$ is sometimes abbreviated to dmerge1$(G_1, v_1, G_2, v_2)$ if we do not attend to the order of the arc sequences.

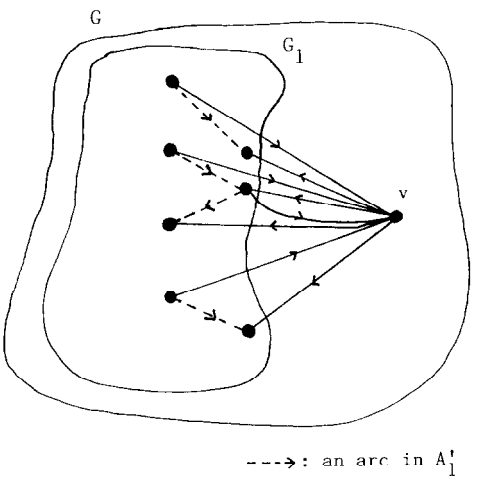



Fig. 12. A digraph composition operation dmerge2$(G_1, A_1', v)$.

**Theorem 4.** *Let digraphs* $G_1=(V_1, A_1)$ *and* $G_2=(V_2, A_2)$ *be independent and k-connected where* $k \geq 2$. *Let* $v_1$ *and* $v_2$ *be vertices in* $V_1$ *and* $V_2$, *respectively, such that* $\mathrm{od}_{G_1}(v_1)=\mathrm{id}_{G_1}(v_1)=\mathrm{od}_{G_2}(v_2)=\mathrm{id}_{G_2}(v_2)=k$. *Then, the digraph* $G=\mathrm{dmerge1}(G_1, v_1, G_2, v_2)$ *is k-connected.*

**Proof.** Let $G=(V, A)$. We shall show that $P_G(u, v) \geq k$ for any distinct vertices $u$ and $v$ in $V$.

(Case 1) $u, v \in V_1 - \{v_1\}$: There exists a set $P$ of internally-disjoint $u$–$v$ paths such that $|P| \geq k$ in $G_1$ because $G_1$ is $k$-connected. In $G_1$, at most one path in $P$ goes through $v_1$.

If there exists no path which goes through $v_1$, then $P_G(u, v) \geq k$ obviously holds.

In the case there exists exactly one path $p$ which goes through $v_1$, $p$ can be denoted by $\langle u, \ldots, w_1, v_1, w_1', \ldots, v\rangle$ where $w_1 \in \mathrm{Fin}(v_1)$ and $w_1' \in \mathrm{Fout}(v_1)$.

By the definition of dmerge1, there exist $w_2$ and $w_2'$ in $V_2 - \{v_2\}$ such that $(w_1, w_2) \in A$ and $(w_2', w_1') \in A$. As $P_{G_2}(w_2, w_2') \geq k$, there exists at least one $w_2$-$w_2'[V_2 - \{v_2\}]$ path. Let $p'$ denote one of them. In $G$, a $u$–$v$ path can be constructed by replacing $\langle w_1, v_1, w_1'\rangle$ in the $u$–$v$ path $p$ by three paths $\langle w_1, w_2\rangle$, $p'$ and $\langle w_2', w_1'\rangle$ where the constructed path is independent of any path in $P - \{p\}$. Therefore, $P_G(u, v) \geq k$ holds.

(Case 2) $u, v \in V_2 - \{v_2\}$: $P_G(u, v) \geq k$ holds by the similar discussion to Case 1.

(Case 3) $u \in V_1 - \{v_1\}$ and $v \in V_2 - \{v_2\}$: At first we show that there exists a $(u, \mathrm{Fin}(v_1))[V_1 - \{v_1\}]$-fanout in $G_1$.

In the case of $u \in V_1 - \{v_1\} - \mathrm{Fin}(v_1)$, it follows from Proposition 3 and $\mathrm{id}(v_1)=k$ that there exists a $(u, \mathrm{Fin}(v_1))[V_1 - \{v_1\}]$-fanout.

In the case of $u \in \mathrm{Fin}(v_1)$, there exists a $(u, \mathrm{Fin}(v_1) \cup \{v_1\} - \{u\})$-fanout from Proposition 3. It follows from $\mathrm{id}(v_1)=k$ that the $u$–$v_1$ path included in the fanout is represented as $\langle u, v_1\rangle$. Therefore, all paths of the fanout whose end vertices are included in $\mathrm{Fin}(v_1) - \{u\}$ do not go through $v_1$, that is, there exists a $(u, \mathrm{Fin}(v_1))[V_1 - \{v_1\}]$-fanout.

Similarly, we can show that there exists $(\mathrm{Fout}(v_2), v)[V_2 - \{v_2\}]$-fanin in $G_2$.

Also, there exists a $(\mathrm{Fin}(v_1), \mathrm{Fout}(v_2))[\mathrm{Fin}(v_1) \cup \mathrm{Fout}(v_2)]$-link in $G$ which consists only of newly added arcs.

Therefore, $k$ internally-disjoint $u$–$v$ paths in $G$ can be constructed from the $(u, \mathrm{Fin}(v_1))$-fanout, the $(\mathrm{Fin}(v_1), \mathrm{Fout}(v_2))$-link and the $(\mathrm{Fout}(v_2), v)$-fanin; hence, $P_G(u, v) \geq k$ holds.

(Case 4) $u \in V_2 - \{v_2\}$ and $v \in V_1 - \{v_1\}$: Similar discussion shows that $P_G(u, v) \geq k$ holds. □

**Theorem 5.** *Let* $A_1'$ *be an arbitrary independent arc set of the digraph* $D_{n,k}$ *such that* $|A_1'|=k$ *and let* $v$ *be any vertex such that* $v \notin V(D_{n,k})$ $(n-3 \geq k \geq 3)$. *Then, the digraph* $G=\mathrm{dmerge2}(D_{n,k}, A_1', v)$ *is k-regular and k-connected.*

**Proof.** It is clear from $|A_1'|=k$ and the definition of dmerge2 that $G$ is $k$-regular.

In what follows, $G$ is showed to be $k$-connected by showing either $P_G(v_1, v_2) \geq k$ or $|C_G(v_1, v_2)| \geq k$ holds for any distinct vertices $v_1$ and $v_2$ such that $(v_1, v_2) \notin A$.

(Case 1) $v_1 = v$: From $(v_1, v_2) \notin A$, $v_2 \notin h(A_1')$. Let FO be a $(v_1, h(A_1'))$-fanout $\{\langle v, h(e')\rangle \mid e' \in A_1'\}$. As $D_{n,k}$ is $k$-connected, there exists a $(h(A_1'), v_2)$-fanin in $D_{n,k}$ termed FI. It follows from $v_2 \notin h(A_1')$ that FI does not use any arc in $A_1'$. Therefore, $k(=|h(A_1')|)$ internally-disjoint $v_1$-$v_2$ paths are constructed in $G$ from FO and FI, that is, $P_G(v_1, v_2) \geq k$ holds.

(Case 2) $v_2 = v$: Similarly $P_G(v_1, v_2) \geq k$ holds.

(Case 3) $v_1 \neq v$ and $v_2 \neq v$: Assume that $|C_G(v_1, v_2)| \leq k-1$. From Lemma 5, $D_{n,k} - A_1'$ is $(k-1)$-connected. Therefore, $C_G(v_1, v_2) \subseteq V(G) - \{v\}$, that is, $v \notin C_G(v_1, v_2)$. On the other hand, there exists a $v_1$-$v_2$ path in $D_{n,k} - C_G(v_1, v_2)$ because $D_{n,k}$ is $k$-connected. Let $p = \langle w_1(=v_1), w_2, \ldots, w_q(=v_2)\rangle$ be the $v_1$-$v_2$ path. Let $I = \{i \mid 1 \leq i \leq q-1 \text{ and } (w_i, w_{i+1}) \in A_1'\}$.

In the case of $I = \emptyset$, it is clear that the $v_1$-$v_2$ path $p$ exists in $G - C_G(v_1, v_2)$, which is a contradiction.

In the case of $I \neq \emptyset$, let $i_1 = \min I$ and $i_2 = \max I$. Then, it follows from $v \notin C_G(v_1, v_2)$ and the definition of dmerge2 that there exist $(w_{i_1}, v)$ and $(v, w_{i_2+1})$ in $G - C_G(v_1, v_2)$. Therefore, the $v_1$-$v_2$ path $\langle w_1(=v_1), w_2, \ldots, w_{i_1}, v, w_{i_2+1}, \ldots, w_q(=v_2)\rangle$ exists in $G - C_G(v_1, v_2)$, which is a contradiction. □

In the rest of this section, a class of $k$-regular and $k$-connected digraphs called $k$-B-digraphs is introduced.

**Definition 12.** For any integer $k$ such that $k \geq 2$, a set of digraphs $BD_k$ is defined to be the minimum set of digraphs which contains $\{D_{n,k} \mid n > k\}$ and is closed under the following rule M1 in the case of $k = 2$ and under the following two rules M1 and M2 in the case of $k \geq 3$.

M1: For any digraphs $B_1 = (V_1, A_1)$ and $B_2 = (V_2, A_2)$ in $BD_k$ and any vertices $v_1 \in V_1$ and $v_2 \in V_2$, the digraph dmerge1$(B_1, v_1, B_2, v_2)$ belongs to $BD_k$.

M2: Let $D_{n,k} = (V, A)$ for any $n$ such that $n - 3 \geq k \geq 3$. For any independent arc set $A'$ $(\subseteq A)$ such that $|A'| = k$ and any vertex $v \notin V$, the digraph dmerge2$(D_{n,k}, A', v)$ belongs to $BD_k$.

A digraph in $BD_k$ is called a *$k$-B-digraph* ($k$-new-Bouquet).

**Theorem 6.** *Any $k$-B-digraph ($k \geq 2$) is $k$-regular and $k$-connected.*

**Proof.** Theorem 6 can be proved by mathematical induction on the number of the rule M1 in the Definition 12 used to obtain the $k$-B-digraph. □

## 6. $k$-VCUAP for $k$-ary trees

This section shows the algorithm to construct a $k$-B-digraph from an arbitrary $k$-ary tree ($k \geq 2$) and that the computation time of the algorithm is optimal within a constant factor.

### 6.1. Algorithm

**KDTB algorithm**
(Algorithm for augmenting a $K$-ary Directed Tree to a $k$-B-digraph)

*Input.* An adjacency list $L$ of a $k$-ary tree $T_{in}=(V_{in}, A_{in}, v_0)$ and an integer $k$ such that $|V_{in}|>k\geq 2$. Suppose that the vertices in $V_{in}$ are denoted by integers $[1\cdots|V_{in}|]$.

*Output.* A $k$-B-digraph $B_{out}=(V_{out}, A_{out})$ constructed from $T_{in}$ by adding arcs. The digraph $B_{out}$ is represented by the following data structure:

```
type
  vertex = 1 .. maxvertices;
  barc = record
           head : vertex;
           flag : boolean
         end;
  BGRAPH = array [vertex, 1 .. maxk] of barc;
var
  Bout : BGRAPH;
```

The constant maxvertices is large enough to represent all vertices including auxiliary vertices which appear during the execution of the algorithm, although the number of vertices increases from $|V_{in}|$ to at most $3|V_{in}|$ (see 6.3).

For a vertex $v$ in $V_{out}$ $(=V_{in})$ and an integer $j$ $(1\leq j\leq k)$, let $B_{out}[v, j]=b$. Then,
$(v, b.\text{head})\in A_{in}\subseteq A_{out}$ if and only if $b.\text{flag}=\text{true}$, and
$(v, b.\text{head})\in A_{out}-A_{in}$ if and only if $b.\text{flag}=\text{false}$.

As $B_{out}$ is a $k$-B-digraph constructed from $T_{in}$ by adding arcs, the arc set

$$\{(v, u)\,|\,B_{out}[v, j].\text{head}=u \text{ and } B_{out}[v, j].\text{flag}=\text{false},\ 1\leq j\leq k,\ 1\leq v\leq|V_{in}|\}$$

is one of the solutions of $k$-VCUAP for $T_{in}$.

*Method.* KDTB algorithm consists of two phases.

**(Phase 1)** Preprocess: Here, $T_{in}$ represented by $L$ is converted into $T_0$ which is in the normal-ordering. The procedure for Phase 1 is omitted.

We use the following data types whose intended meanings are explained below.

```
type
  arc = record
          t, h : vertex
        end;
  TREE = record
           odeg, size : array [vertex] of integer;
           s:array [vertex, 1 .. maxk] of vertex
         end;
```

**var**
$T_0$: TREE;

For a variable $e$ of type arc,
$e$ stands for an arc $(u, v)$ if and only if $e.t = u$ and $e.h = v$.
For a variable $T_0$ of type TREE,
$T_0.\mathrm{odeg}[v]$ stands for $\mathrm{od}(v)$,
$T_0.\mathrm{size}[v]$ stands for $n(v)$; that is, the number of vertices in the subtree with the root $v$, and
$T_0.s[v, j]$ $(1 \leq j \leq \mathrm{od}(v))$ stands for $v^j$; that is, the $j$-th son of $v$.

**(Phase 2)** A $k$-B-digraph $B_{\mathrm{out}}$ is constructed from $T_0$: By the following procedure bmake$(T_0, v_0, k, B_{\mathrm{out}})$, $B_{\mathrm{out}} \in BD_k$ is constructed from $T_0$.

The overall outline of the procedure bmake is that the given $k$-ary tree is divided into some smaller trees in the first place so that each of them can be augmented to a super daisy chain. This division of the given $k$-ary tree is made in the steps (3)–(14) and/or the step (31) of the procedure bmake described below. Next, each of the trees is augmented to a super daisy chain by adding arcs in the steps (23)–(24), (26)–(27), (29)–(30) or (32)–(36). Lastly, in the step (19) and/or (37)–(38), the super daisy chains are merged into a $k$-B-digraph by the digraph composition operations dmerge1 and dmerge2 so that the deleted arcs to divide the initial tree are rejoined. Recall that these digraph composition operations preserve $k$-regularity and $k$-connectivity.

In what follows, the procedure bmake is described in the Pascal-like language. For readability, some graph-theoretical notations are used in the description. Especially, the procedure update_*Bs*(*B*) has the parameter *B* which is a digraph in the graph-theoretical notation, and updates only the related part of the variable *Bs* of type BGRAPH to store the structure of *B*. In this way, the procedure links the graph theoretical notation with the concrete data representation in the program. Note that update_*Bs*(*B*) dose not influence any digraph which has no vertex in common with the digraph *B*, although *Bs* stores structures of some digraphs during the execution of the procedure bmake.

**procedure** bmake(*Ts*: TREE; $v$: vertex; *kl*: integer; **var** *Bs*: BGRAPH);
{Initially *Ts* of type TREE represents a tree with the root $v$, and it will represent some ordered trees obtained from the initial tree in the execution of the algorithm.}
**var**
last: integer;
{The largest integer which stands for a vertex.}
**procedure** submake($v_1, v_2$: vertex);
{A *kl*-B-digraph is constructed from the tree $T$ with the root $v_1$. Here, $v_2$ represents a root of a subtree which includes all *kl*-bridges in $T$.}
**var**

```
    brd : arc;
    w1, w2, vc : vertex;
    sn : integer;
  procedure make_daisy(W0 : vertex set; g : order W0 → [0 ··· |W0| − 1]);
    {A super daisy chain with the vertex set W0 is constructed.}
    begin
      let B = (W0, {(u, v) | 0 < g(v) − g(u) ≤ kl});
      update_Bs(B)
    end; {make_daisy}
  procedure dmerge_1(u1, u2 : vertex; e : arc);
    {A kl-B-digraph is constructed by the rule M1.}
    begin
      letting G1 (resp. G2) be a kl-B-digraph which includes u1 (resp. u2), let
      B = dmerge1(G1, u1, G2, u2) where ⟪IA(u1)⟫1 = (e . t, u1) and ⟪OA(u2)⟫1 =
      (u2, e . h) hold to certify that the arc e is rejoined;
        {dmerge1 is the digraph operation introduced in 5.}
      update_Bs(B)
    end; {dmerge_1}
  procedure dmerge_2(W1 : vertex set; Ad1 : arc set; u : vertex);
    {A kl-B-digraph is constructed by the rule M2.}
    begin
      letting G1 be the digraph whose vertex set is W1, let B = dmerge2(G1, Ad1, u);
        {dmerge2 is the digraph operation introduced in 5.}
      update_Bs(B)
    end; {dmerge_2}

    begin {submake}
(1)   let T = (V, A, v1) be the tree with the root v1 in Ts;
(2)   if there exists a kl-bridge in the rightmost path of T(v2) then begin
        {A kl-B-digraph is constructed by the rule M1.}
(3)     set brd to the kl-bridge which is nearest to v2 in the rightmost path of
        T(v2);
(4)     sn := Ts . odeg[brd . t]; {Ts . s[brd . t, sn] = brd . h}
(5)     w1 := last + 1; w2 := last + 2; last := last + 2;
          {w1 and w2 stand for newly introduced vertices.}
(6)     with Ts do begin
(7)       s[brd . t, sn] := w1;
(8)       odeg[w1] := 0;
(9)       size[w1] := 1;
(10)      s[w2, 1] := brd . h;
(11)      odeg[w2] := 1;
            {By the steps (7)–(11), T is divided into two trees.}
(12)      let trees T1 = (V1, A1) and T2 = (V2, A2) be as follows:
```

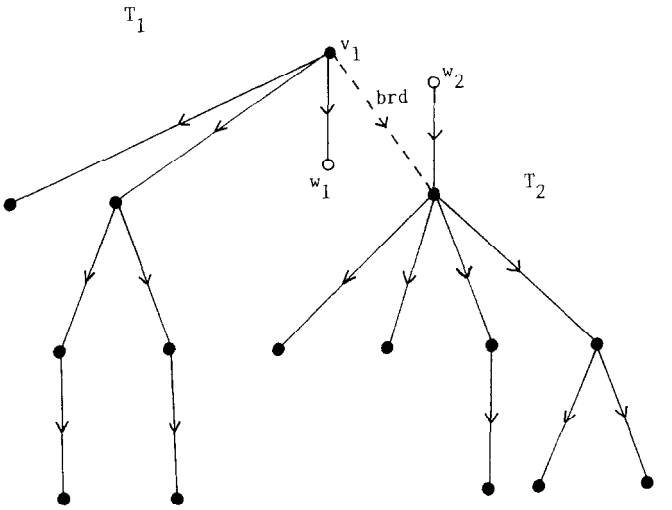



Fig. 13. The $kl$-ary trees $T_1$ and $T_2$ produced in the steps (7)–(11). ($kl=4$.)

{Fig. 13}
$V_1=(V-V(brd.h))\cup\{w_1\}$
$A_1=A(T[V-V(brd.h)])\cup\{(brd.t, w_1)\}$
$V_2=V(brd\ h)\cup\{w_2\}$
$A_2=A(T[V(brd.h)])\cup\{(w_2, brd.h)\}$;

(13) size[$v_1$] := size[$v_1$] − size[$brd.h$] + 1;

(14) size[$w_2$] := size[$brd.h$] + 1
{size[$v_1$] and size[$w_2$] are set to $|V_1|$ and $|V_2|$, respectively. They are used in order to find $kl$-bridges in (2) and $(kl-1)$-weak-centroid-vertices in (21).}

**end**;

(15) arrange $T_1(brd.t)$ in the normal-ordering;
{$T_2$ is already in the normal-ordering.}

(16) submake($v_1$, $brd.t$);
{$T_1$ is augmented to a $kl$-B-digraph.}

(17) submake($w_2$, $w_2$);
{$T_2$ is augmented to a $kl$-B-digraph.}

(18) let $B_1$ (resp. $B_2$) be the $kl$-B-digraph constructed from $T_1$ (resp. $T_2$);

(19) dmerge_1($w_1$, $w_2$, $brd$);
{A $kl$-B-digraph is constructed from $B_1$ and $B_2$ by the operation dmergel. Note that the arc $brd$ deleted in the step (7) is rejoined.}

**end**

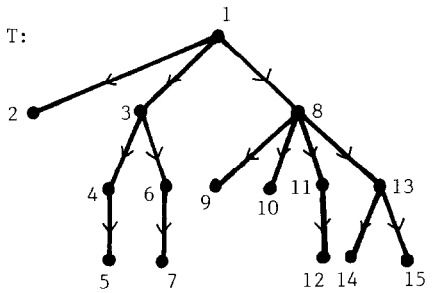
**else** {There is no $kl$-bridge} **begin**

(20) reform $T$ into the normal-ordering;

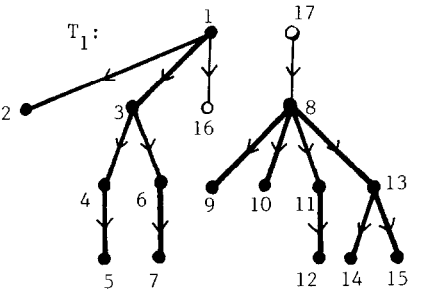
(21) set $vc$ to a $(kl-1)$-weak-centroid-vertex in the rightmost path of $T$;

```
(22)      if Ts.odeg[vc] ≤ 2 then begin
            {T is augmented to D_{|V|,kl} by adding arcs.}
(23)        find the depth-first-order f: V → [0 ··· |V| − 1];
(24)        make_daisy(V, f)
          end
(25)      else if (the number of leaves in T) ≤ kl then begin
            {T is augmented to D_{|V|,kl} by adding arcs.}
(26)        find the breadth-first-order f: V → [0 ··· |V| − 1];
(27)        make_daisy(V, f)
          end
(28)      else if |V| = kl + 3 then begin
            {T is augmented to D_{|V|,kl} by adding arcs.}
(29)        find the depth-first-order f: V → [0 ··· |V| − 1];
(30)        make_daisy(V, f)
          end
          else begin
            {A kl-B-digraph is constructed by the rule M2.}
(31)        let T_3 = (V_3, A_3) be the digraph T[V − {vc}];
              {T_3 is augmented to B_3 = D_{|V_3|,kl}.}
(32)        find the depth-first-order f_0: V_30 → [0 ··· |V_30| − 1] for T[V − V(vc)] =
            (V_30, A_30);
(33)        find the depth-first-order f_j: V_3j → [0 ··· |V_3j| − 1] for T(vc^j) = (V_3j, A_3j)
            (1 ≤ j ≤ od_T(vc));
(34)        set f: V_3 → [0 ··· |V_3| − 1] as follows:
              f(u) := f_i(u) + Σ_{j=0}^{i−1} |V_3j| for u ∈ V_3i;
(35)        make_daisy(V_3, f);
(36)        let B_3 = (V_3, A_3') be D_{|V_3|,kl} constructed in the step (35);
(37)        find an independent arc set Ad (⊆ A_3' − A_3) such that
              |Ad| = kl,
              Ad ⊇ {(vc', vc^1)} ∪ {e ∈ A_1(B_3) | h(e) = vc^i (2 ≤ i ≤ od(vc))}
              and Ad − {(vc', vc^1)} ⊆ A_1(B_3)
                (vc' is the father of vc if vc ≠ v_1, (vc', vc^1) ∈ A_1(B_3) if vc = v_1);
              {Recall that the arc set A_1(B_3) of a super daisy chain B_3 is defined
               in Definition 8.}
(38)        dmerge_2(V_3, Ad, vc)
          end
        end; {submake}
      begin {bmake}
(39)    last := (the number of vertices of the initial tree);
(40)    submake(v, v)
      end; {bmake}
```

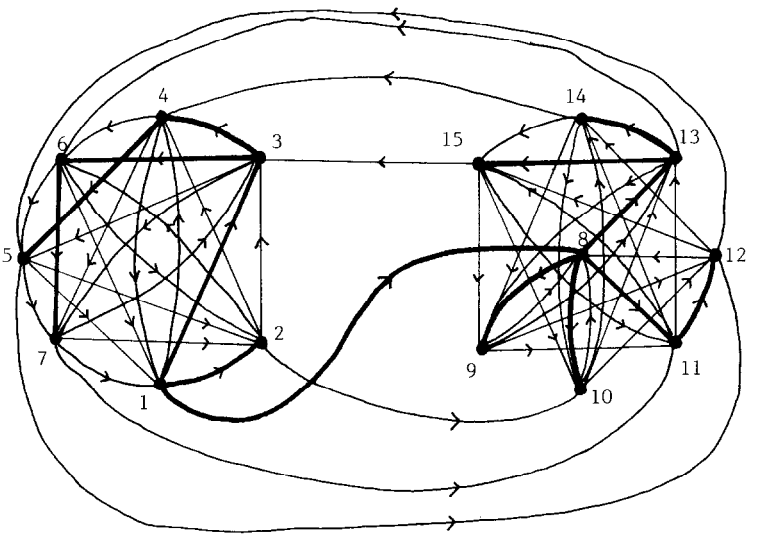
**Example 8.** Fig. 14 shows the process of augmentation of a 4-ary tree to a 4-B-digraph by KDTB algorithm.




(a) A given 4-ary tree T.




(b) Division of T.




(c) A 4-B-digraph constructed from T.

Fig. 14. The process of augmentation of a 4-ary tree to a 4-B-digraph by KDTB algorithm.

## 6.2. *Correctness of KDTB algorithm*

Let $B=(V_B, A_B) \in BD_{kl}$ be a digraph which is constructed from the tree $T=(V, A, v_1)$ by the procedure submake.

To show the correctness of the KDTB algorithm, we use mathematical induction on the number of calls of submake caused in the step (16) to show that $B$ satisfies the following condition **C**.

**C**: $B \in BD_{kl}$ and $V_B = V$ and $A_B \supseteq A$.

Before proving that $B$ satisfies the condition **C**, it is showed that there is no tree which causes infinite repetition of calls of submake.

### *6.2.1. Finiteness of repetition of calls of submake*

**Theorem 7.** *There is no tree which causes infinite repetition of calls of submake.*

**Proof.** From the definition of a $kl$-bridge, the $kl$-bridge $brd$ satisfies $|V - V(brd.h)| \geq kl$ and $|V(brd.h)| \geq kl$, where $kl$ $(=k) \geq 2$. Let $T_1 = (V_1, A_1)$ and $T_2 = (V_2, A_2)$ be trees obtained from the steps (7)-(11). Then, $|V_1| \leq |V| - kl + 1 \leq |V| - 1$ holds from $V_1 = (V - V(brd.h)) \cup \{w_1\}$. Similarly, $|V_2| \leq |V| - 1$ holds. Therefore, the number of vertices in the tree for which submake is applied decreases and there is no $kl$-bridge after repeating the calls of submake. This fact means that the calls of submake do not continue infinitely. □

### *6.2.2. The basis to show B satisfies the condition* **C**

As the basis of mathematical induction, we shall show that $B$ constructed in the steps (24), (27), (30) and (38) in the procedure submake without calling submake satisfies the condition **C**.

At first, we shall show that any tree $T$ in the step (20) in the procedure submake has no $kl$-bridge. This property is proved in the series of the following two lemmas.

**Lemma 6.** *For a kl-ary tree T with kl-bridges, let* $v_2$ *be a vertex such that all kl-bridges of T are in* $T(v_2)$ *and let* $T(v_2)$ *be in the normal-ordering. Then, there exists the kl-bridge e in the rightmost path of* $T(v_2)$ *such that all kl-bridges are in* $T(t(e))$.

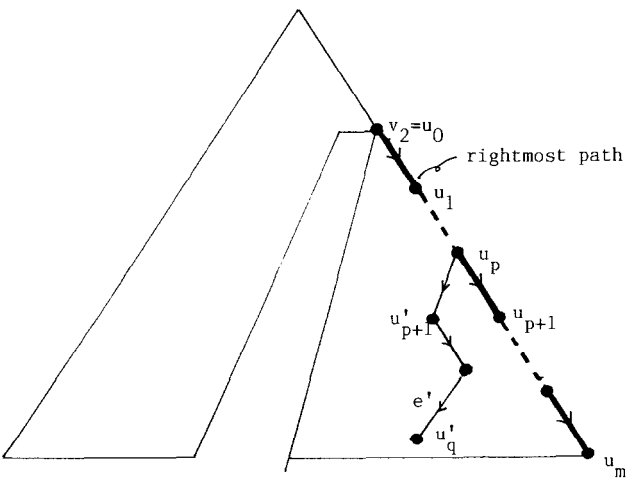



Fig. 15. A $kl$-bridge $(u_p, u_{p+1})$ in the rightmost path in case $e'$ is a $kl$-bridge.

**Proof.** Let $\langle u_0(=v_2), u_1, \ldots, u_m\rangle$ be the rightmost path of $T(v_2)$. Suppose there exists a $kl$-bridge $e'$ in $T(v_2)$ but the rightmost path. Let

$$\langle u_0(=v_2), u_1, \ldots, u_p, u'_{p+1}, \ldots, u'_q(=h(e'))\rangle$$

be the $v_2-h(e')$ path where $u'_{p+1}\neq u_{p+1}$ (Fig. 15). It follows from

$$n(u_{p+1})\geq n(u'_{p+1})\geq n(h(e'))\geq kl, \quad \text{and} \quad |V(T)|-n(u_{p+1})\geq n(u'_{p+1})\geq kl$$

that $(u_p, u_{p+1})$ is a $kl$-bridge: that is, there exists a $kl$-bridge $(u_p, u_{p+1})$ in the rightmost path such that $e'$ is included in $T(u_p)$. Therefore, letting $e$ be the $kl$-bridge $(u_i, u_{i+1})$ such that $(u_j, u_{j+1})$ is not a $kl$-bridge for each $j$ $(0\leq j<i)$, all $kl$-bridges are in $T(t(e))$. □

**Lemma 7.** *Any tree $T$ in the step* (20) *in the procedure submake that has no kl-bridge in the rightmost path of $T(v_2)$ has no kl-bridge.*

**Proof.** If $v_2$ is the vertex such that all $kl$-bridges of $T$ are in $T(v_2)$, then the lemma holds from Lemma 6.

When submake is called in the step (40), $v_2$ is set to the root of $T$, so that all $kl$-bridges of $T$ are in $T(v_2)$. The same holds for the call of submake in the step (17).

From Lemma 6 and the fact that *brd* set in the step (3) is the $kl$-bridge which is nearest to $v_2$ in the rightmost path of $T(v_2)$, all $kl$-bridges of $T$ are in $T(brd.t)$. By the steps (7)–(11), $T$ is divided into $T_1$ and $T_2$, and a $kl$-bridge of $T_1$ is also that of $T$. Therefore, all $kl$-bridges of $T_1$ are in $T_1(brd.t)$ and the lemma holds for the call of submake in the step (16). □

Now we show that any tree $T$ in the step (21) has a $(kl-1)$-weak-centroid-vertex in the rightmost path.

**Lemma 8.** *For any tree $T$ with no kl-bridge and $|V(T)|>kl$, there exists a $(kl-1)$-weak-centroid-vertex in the rightmost path in the normal-ordering.*

**Proof.** There exists the vertex $vc$ in the rightmost path in the normal-ordering such that $n(vc)\geq kl$ and $n(vc^{\mathrm{od}(vc)})\leq kl-1$ hold. For each $i$ $(1\leq i\leq \mathrm{od}(v))$, $n(vc^i)\leq n(vc^{\mathrm{od}(vc)})\leq kl-1$ holds.

In the case where $vc$ is the root $v_1$ of $T$, $vc$ is obviously a $(kl-1)$-weak-centroid-vertex.

In the case of $vc\neq v_1$, there exists the father $v'$ of $vc$. As $n(vc)\geq kl$ holds and there exists no $kl$-bridge in $T$, $|V|-n(vc)\leq kl-1$ holds. Therefore, $vc$ is a $(kl-1)$-weak-centroid-vertex. □

Now we show that $B$ constructed in the steps (24), (27), (30) and (38) satisfies the condition **C**.

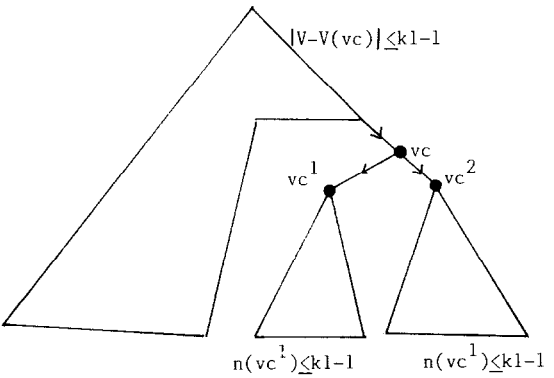



Fig. 16. A tree $T$ from which $B$ is constructed in the step (24).

**Lemma 9.** *$B$ constructed in the step* (24) *from a $kl$-ary tree $T=(V,A)$ with no $kl$-bridge such that* $\mathrm{od}(vc)\leq 2$ *for the $(kl-1)$-weak-centroid-vertex $vc$ satisfies the condition* **C**.

**Proof.** It suffices to show that the depth-first-order $f: V\to[0\cdots|V|-1]$ found in the step (23) is a $kl$-strong-bandwidth-order. It is clear that $f(u_2)-f(u_1)>0$ holds for any arc $(u_1,u_2)\in A$ because $f$ is the depth-first-order. It follows from $|V-V(vc)|\leq kl-1$, $n(vc^1)\leq kl-1$ and $n(vc^2)\leq kl-1$ (if $\mathrm{od}(vc)=2$) that $f(u_2)-f(u_1)\leq kl$ holds for any arc $(u_1,u_2)\in A$ (Fig. 16).

Consequently, $f$ is a $kl$-strong-bandwidth-order. □

**Lemma 10.** *$B$ constructed in the step* (27) *from a $kl$-ary tree $T=(V,A)$ with no $kl$-bridge such that the number of leaves is at most $kl$ satisfies the condition* **C**.

**Proof.** It suffices to show that the breadth-first-order $f: V\to[0\cdots|V|-1]$ found in the step (26) is a $kl$-strong-bandwidth-order. It is clear that $f(u_2)-f(u_1)>0$ holds for any arc $(u_1,u_2)\in A$. In the following, it is showed that $f(u_2)-f(u_1)\leq kl$ holds for any arc $(u_1,u_2)\in A$.

Assume that $f(x_2)-f(x_1)>kl$ holds for some arc $(x_1,x_2)\in A$. Let $p=\mathrm{depth}(x_1)$; then $\mathrm{depth}(x_2)=p+1$. For each $i$ $(1\leq i\leq f(x_2)-f(x_1))$, let $y_i$ be the vertex such that $f(y_i)=f(x_1)+i$. Note that $y_{f(x_2)-f(x_1)}$ is $x_2$. Let $q$ be the integer such that $\mathrm{depth}(y_q)=p$ and $\mathrm{depth}(y_{q+1})=p+1$ (Fig. 17). From the definition of the breadth-first-order and the fact that $(x_1,x_2)\in A$, $(y_i,y_j)$ does not belong to $A$ for any $i$ and $j$ such that $1\leq i\leq q<j\leq f(x_2)-f(x_1)$. Therefore, for any $i$ and $j$ $(1\leq i<j\leq f(x_2)-f(x_1))$, any leave in the subtree with the root $y_i$ is distinct from the one in the subtree with the root $y_j$. Therefore, there exist at least $f(x_2)-f(x_1)$ leaves, that is, there exist more than $kl$ leaves. This means that the assumption that $f(x_2)-f(x_1)>kl$ holds for some arc $(x_1,x_2)\in A$ contradicts the fact that there exist at most $kl$ leaves. Consequently, $f(u_2)-f(u_1)\leq kl$ holds for any arc $(u_1,u_2)\in A$. □

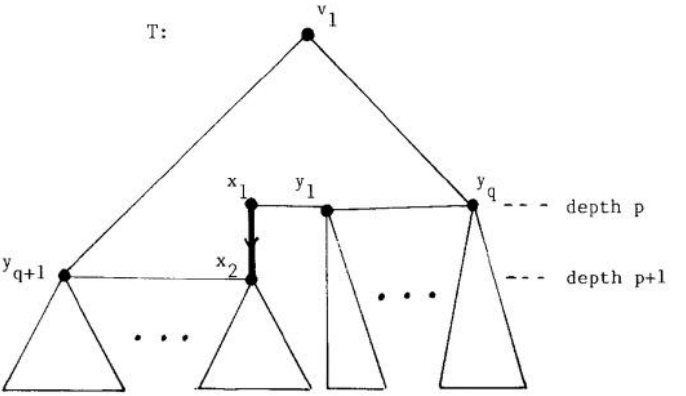



Fig. 17. Subtrees with the roots $x_1$, $x_2$ and $y_i$ $(i=1,\ldots,f(x_2)-f(x_1))$.

**Lemma 11.** *B constructed in the step* (30) *from a kl-ary tree* $T=(V,A)$ *with no kl-bridge such that* $|V|=kl+3$ *satisfies the condition* **C**.

**Proof.** It suffices to show that the depth-first-order $f: V\to[0\cdots|V|-1]$ found in the step (29) is a *kl*-strong-bandwidth-order. It is clear that $f(u_2)-f(u_1)>0$ holds for any arc $(u_1,u_2)\in A$.

There exist two internal vertices and $kl+1$ leaves in $T$ from the fact that (the number of leaves)$>kl$, $|V|=kl+3$ and $T$ is a *kl*-ary tree. From $\mathrm{od}(v_1)\le kl$ and $\mathrm{od}(v_1^{\mathrm{od}(v_1)})\le kl$, it is clear that $f(u_2)-f(u_1)\le kl$ holds for any arc $(u_1,u_2)\in A$. □

**Lemma 12.** $B_3=(V_3,A_3')$ *constructed in the step* (35) *from* $T_3=(V_3,A_3)=T[V-\{vc\}]$ *is* $D_{|V_3|,kl}$ *and satisfies* $V_3=V-\{vc\}$ *and* $A_3'\supseteq A_3=A-(\mathrm{IA}(vc)\cup\mathrm{OA}(vc))$.

**Proof.** It is evident that $V_3=V-\{vc\}$ holds. It suffices to show that $f: V_3\to[0\cdots|V_3|-1]$ found in the step (34) is a *kl*-strong-bandwidth-order of $T_3$.

It is clear that $f(u_2)-f(u_1)>0$ holds for any arc $(u_1,u_2)\in A_3$. As $vc$ is a $(kl-1)$-weak-centroid-vertex, $|V-V(vc)|<kl$ and $n(vc^i)<kl$ for each $i$ $(1\le i\le\mathrm{od}(vc))$ hold. Therefore, $f(u_2)-f(u_1)\le kl$ holds for any arc $(u_1,u_2)\in A(T_3)$. □

**Lemma 13.** *B constructed in the step* (38) *by dmerge_2 from* $B_3$ *and* $vc$ *satisfies the condition* **C**.

**Proof.** Before proving the lemma, it is showed that the arc set $Ad$ in the step (37) can be found in $B_3$.

Let $Ad'=\{e\in A_1(B_3)\mid t(e)$ is a leaf of $T$ and $h(e)\ne vc^1\}$ (where $B_3$ is a super daisy chain and the arc set $A_1(B_3)$ is defined in Definition 8); then $Ad'$ is independent and $Ad'\subseteq A_1(B_3)-A$. Moreover, $|Ad'|\ge kl$ holds because there exist at least $kl+1$ leaves in $T$. The arc $(vc',vc^1)$ is independent of any arc in $Ad'$, which is shown by

the following (i) and (ii).

(i) In case of $vc \neq v_1$, $vc'$ is the father of $vc$ as determined by the step (37). That is, $vc'$ is not a leaf of $T$. Therefore, letting $e$ be any arc in $Ad'$, $t(e)$ is not $vc'$ and it is clear from the definition of $Ad'$ that $h(e)$ is not $vc^1$. It follows that the arc $(vc', vc^1)$ is independent of any arc in $Ad'$.

(ii) In case of $vc = v_1$, then $(vc', vc^1) \in A_1(B_3)$ holds by the step (37). Since the arc set $A_1(B_3)$ is independent and $Ad'$ is a subset of $A_1(B_3) - \{(vc', vc^1)\}$, the arc $(vc', vc^1)$ is independent of any arc in $Ad'$.

Let $Ad_1 = \{(vc', vc^1)\} \cup \{e \in A_1(B_3) \mid h(e) = vc^i \ (2 \leq i \leq \mathrm{od}(vc))\}$ and let $Ad_2$ be any subset of $Ad' - Ad_1$ such that $|Ad_2| = kl - |Ad_1|$. By the following consideration, we can show that $Ad_1 - \{(vc', vc^1)\}$ $(= \{e \in A_1(B_3) \mid h(e) = vc^i \ (2 \leq i \leq \mathrm{od}(vc))\}) \subseteq Ad'$ holds. Letting $e$ be any arc such that $e \in A_1(B_3)$ and $h(e) = vc^i$ where $2 \leq i \leq \mathrm{od}(vc)$, $t(e)$ is a leaf of $T(vc^{i-1})$, because $t(e)$ is the vertex of $T(vc^{i-1})$ with $f_{i-1}(t(e)) = \max\{f_{i-1}(v) \mid v \in V(vc^{i-1})\}$ from the definition of $f$ in the step (34). (Recall that $f_{i-1}$ is the depth-first-order of $T(vc^{i-1})$.)

Letting $Ad = Ad_1 \cup Ad_2$, we will show that $Ad$ satisfies the following conditions described in the step (37).

(a) $Ad$ is independent.

(b) $Ad \subseteq A_3' - A_3$.

(c) $|Ad| = kl$.

(d) $Ad \supseteq \{(vc', vc^1)\} \cup \{e \in A_1(B_3) \mid h(e) = vc^i \ (2 \leq i \leq \mathrm{od}(vc))\}$.

(e) $Ad - \{(vc', vc^1)\} \subseteq A_1(B_3)$.

It is clear from the definition of $Ad$ that the conditions (b), (c) and (d) hold. It follows from $Ad_1 - \{(vc', vc^1)\} \subseteq Ad' \subseteq A_1(B_3) - A$ that the condition (e) holds. The arc set $Ad$ is independent because $Ad - \{(vc', vc^1)\}$ is a subset of the independent arc set $Ad'$ and the arc $(vc', vc^1)$ is independent of any arc in $Ad'$. Thus, the condition (a) holds.

Now let us prove the lemma itself.

The digraph $B_3$ *is* $D_{|V_3|, kl}$ from Lemma 12 and it is clear that $|V_3| \geq kl + 3$, $kl\ (\geq \mathrm{od}(vc)) \geq 3$, $|Ad| = kl$ and $Ad$ is independent. Therefore, $B$ constructed in the step (38) belongs to $BD_{kl}$ from the definition of the $kl$-B-digraph.

Also, it is clear that $V_B = V_3 \cup \{vc\} = V$.

From Lemma 12 and the definition of $Ad$, $A_3' - Ad \supseteq A_3 = A - (\mathrm{IA}(vc) \cup \mathrm{OA}(vc))$. It follows from $Ad \supseteq \{(vc', vc^1)\} \cup \{e \mid h(e) = vc^i \ (2 \leq i \leq \mathrm{od}(vc))\}$ that all arcs in $\mathrm{IA}(vc) \cup \mathrm{OA}(vc)$ are added by dmerge_2$(V_3, Ad, vc)$. Therefore, $A_B \supseteq A$ holds. □

### *6.2.3. Inductive step to show B satisfies the condition* **C**

Assume that for $B$ constructed after at most $m$ $(\geq 0)$ calls of the procedure submake in the step (16), $B$ satisfies the condition **C**. Then, it is shown that $B$ constructed in the step (19) by dmerge_1 after $m+1$ calls of submake satisfies the condition **C**.

**Lemma 14.** *Under the assumption that* $B_1$ *and* $B_2$ *constructed in the step* (16) *and*

(17) *satisfy the condition* **C**, *B constructed in the step* (19) *by dmerge_1 satisfies the condition* **C**.

**Proof.** Let $T_1=(V_1, A_1)$ and $T_2=(V_2, A_2)$, which are constructed in the steps (7)-(11). Let $B_1=(V_1', A_1')$ and $B_2=(V_2', A_2')$, which are constructed in the steps (16) and (17), respectively. From the assumption, $B_1 \in BD_{kl}$, $V_1'=V_1$ and $A_1' \supseteq A_1$ hold. Similarly, $B_2 \in BD_{kl}$, $V_2'=V_2$ and $A_2' \supseteq A_2$ hold.

It is evident from the definition of the *kl*-B-Digraph that $B$ constructed in the step (19) belongs to $BD_{kl}$. It follows from $V_1=(V-V(brd.h))\cup\{w_1\}$ and $V_2=V(brd.h)\cup\{w_2\}$ that $V_B=(V_1\cup V_2)-\{w_1, w_2\}=V$. From the way of dividing $T$ into $T_1$ and $T_2$, $A_1'\cup A_2' \supseteq A_1\cup A_2 \supseteq A-\{brd\}$. Also, from the definition of the digraph conmposition operation dmerge1,

$$A_B=(A_1'\cup A_2'-(\mathrm{IA}(w_1)\cup\mathrm{OA}(w_1)\cup\mathrm{IA}(w_2)\cup\mathrm{OA}(w_2)))$$
$$\cup\{(t(\langle\!\langle\mathrm{IA}(w_1)\rangle\!\rangle_i), h(\langle\!\langle\mathrm{OA}(w_2)\rangle\!\rangle_i)) \mid 1\le i\le kl\}$$
$$\cup\{(t(\langle\!\langle\mathrm{IA}(w_2)\rangle\!\rangle_i), h(\langle\!\langle\mathrm{OA}(w_1)\rangle\!\rangle_i)) \mid 1\le i\le kl\}.$$

It follows from $w_1 \notin V$ and $w_2 \notin V$ that $\mathrm{IA}(w_1)\cup\mathrm{OA}(w_1)\cup\mathrm{IA}(w_2)\cup\mathrm{OA}(w_2)$ does not include any arc in $A$. Therefore, $A_B \supseteq A$ from

$$brd=(t(\langle\!\langle\mathrm{IA}(w_1)\rangle\!\rangle_1), h(\langle\!\langle\mathrm{OA}(w_2)\rangle\!\rangle_1)).$$

Consequently, $B$ constructed in the step (19) satisfies the condition **C**. □

**Theorem 8.** *Let* $B_{\mathrm{out}}=(V_{\mathrm{out}}, A_{\mathrm{out}})$ *be the output of the KDTB algorithm for any integer* $k$ ($k\ge 2$) *and any k-ary tree* $T_{\mathrm{in}}=(V_{\mathrm{in}}, A_{\mathrm{in}})$. *Then, an arc set* $A_{\mathrm{out}}-A_{\mathrm{in}}$ is one of solutions of k-VCUAP for $T_{\mathrm{in}}$.

**Proof.** It follows from Lemmas 9, 10, 13 and 14 that $B_{\mathrm{out}}$ produced by bmake$(T_0, v_0, k, B_{\mathrm{out}})$ satisfies $B_{\mathrm{out}} \in BD_k$, $V_{\mathrm{out}}=V(T_0)=V_{\mathrm{in}}$ and $A_{\mathrm{out}} \supseteq A(T_0)=A_{\mathrm{in}}$. As $B_{\mathrm{out}}$ is $k$-regular and $k$-connected, it follows from Corollary 1 that $A_{\mathrm{out}}-A_{\mathrm{in}}$ is one of the solutions of $k$-VCUAP for $T_{\mathrm{in}}$. □

### *6.3. Computation time of KDTB algorithm*

**Theorem 9.** *The time-complexity of* $k$-VCUAP ($k\ge 2$) *for every k-ary tree* $T=(V, A)$ *is* $\theta(k|V|)$.

**Proof.** (Upper bound): We shall show that the computation time, denoted by time$(T, k)$, of the KDTB algorithm for $T$ and $k$ is $O(k|V|)$.

Let time$_i(T, k)$ stand for the computation time of the Phase $i$ ($i=1, 2$).

(Phase 1): It takes $O(|V|)$ time to find $n(v)$ for each vertex $v$ in $T$ by the postorder traversal of the vertices of $T$ [1]. For some vertex $v$, it takes $O(\mathrm{od}(v)\log \mathrm{od}(v))$ time to sort its sons for normal-ordering. It follows from $\sum_{v\in V}\mathrm{od}(v)=|V|-1$ and

$\mathrm{od}(v) \leq k$ that it takes $O(|V| \log k)$ time to arrange $T$ into normal-ordering. Therefore,

$$\mathrm{time}_1(T, k) = O(|V| \log k) \tag{4}$$

holds.

(Phase 2): Let $r$ be the number of times of dividing $T$ into $T_1$ and $T_2$ by the steps (7)–(11). When a tree is divided in (7)–(11), the number of vertices increases by two and the number of arcs increases by one. Therefore, after dividing a tree $r$ times, the number of vertices and that of arcs become $|V|+2r$ and $|A|+r$, respectively, and $r+1$ $k$-ary trees are produced. On the other hand, each of the $r+1$ $k$-ary trees has at least $k+1$ vertices. Therefore, $r \leq |V|/(k-1)$ follows from $(|V|+2r)/(r+1) \geq k+1$. Consequently, the number of vertices and that of arcs become at most $|V|+2|V|/(k-1)$ and $|A|+|V|/(k-1)$, respectively. Note that $|A| = |V|-1$ holds.

The computation time of the steps (1)–(3) throughout the execution of the algorithm is $O(|V|)$. Because each arc is examined at most once whether or not it is a $k$-bridge and at most $(|V|-1)+2|V|/(k-1)$ arcs appear in the algorithm.

(Case 1) There is no calling of submake (steps (20)–(38)): By the method similar to that of Phase 1, the step (20) is executed in $O(|V| \log k)$ time. It apparently takes $O(|V|)$ time to execute the step (21). It also takes $O(|V|)$ time to find the depth-first-order or the breadth-first-order of $T$ and it takes $O(k|V|)$ time to execute make_daisy$(V, f)$ in the steps (24), (27) or (30). Therefore, it takes $O(k|V|)$ time to execute the steps (22)–(30). The steps (31)–(34) can be executed in $O(|V|)$ time. It takes $O(k|V|)$ time and $O(|V|)$ time to execute the step (35) and the steps (36)–(37), respectively. From $|Ad| = k$, it takes $O(k)$ time to execute the step (38). Therefore,

$$\mathrm{time}_2(T, k) = O(k|V|). \tag{5}$$

From the equations (4) and (5),

$$\mathrm{time}(T, k) = \mathrm{time}_1(T, k) + \mathrm{time}_2(T, k) = O(k|V|). \tag{6}$$

(Case 2) There are calls of submake (steps (4)–(19)): It takes constant time to execute the steps (4)–(14), and it takes $O(k)$ time to arrange $T_1(brd.t)$ in the normal-ordering in the step (15).

Let $\mathrm{time}'(T_1, k)$ and $\mathrm{time}'(T_2, k)$ be the computation time (except for the steps (1)–(3)) to construct $B_1$ from $T_1$ and the one to construct $B_2$ from $T_2$. As the procedure dmerge_1 can be executed in $O(k)$ time,

$$\mathrm{time}(T, k) = \mathrm{time}_1(T, k) + O(|V|) + \mathrm{time}'(T_1, k) + \mathrm{time}'(T_2, k) + O(k). \tag{7}$$

Let $Ts_i = (Vs_i, As_i)$ $(0 \leq i \leq r)$ be each of $r+1$ trees constructed by $r$ executions of steps (7)–(11). From the equations (4), (6) and (7),

$$\mathrm{time}(T, k) = O(|V| \log k) + O(|V|) + \sum_{i=0}^{r} O(k|Vs_i|) + \sum_{i=0}^{r} O(k)$$

holds. From $\sum_{i=0}^{r} |Vs_i| \leq |V| + 2|V|/(k-1)$ and $r \leq |V|/(k-1)$, $\mathrm{time}(T, k) = O(k|V|)$.

In constructing $B \in BD_k$ from $T$, for each arc $e$, $e$.flag is manipulated as follows:

$e$.flag is true if and only if $e \in A$.

Therefore, an arc set $A(B)-A$ is obtained from $B$ in $O(k|V|)$ time.

Consequently, $k$-VCUAP for any $k$-ary tree $T=(V,A)$ can be solved in $O(k|V|)$ time by the KDTB algorithm.

(Lower bound): From Theorem 1, the solution of $k$-VCUAP for any $k$-ary tree $T=(V,A)$ includes at least $k|V|-|A|=(k-1)|V|+1$ arcs. Therefore, the lower bound of the computation time of $k$-VCUAP for any $k$-ary tree $T=(V,A)$ is $\Omega(k|V|)$. □

## 7. $k$-VCUAP for directed trees

This section presents the algorithm, called DT algorithm, to solve $k$-VCUAP ($k \geq 2$) for general trees in the optimal time within a constant factor. The KDTB algorithm in the previous section is used as part of the DT algorithm.

### 7.1. Algorithm

**DT algorithm**
(Algorithm for augmenting a Directed Tree to a $k$-connected digraph)

*Input.* An adjacency list $L$ of a tree $T=(V,A,v_0)$ and an integer $k$ ($|V|>k\geq 2$).

*Output.* A $k$-connected digraph $B=(V_B,A_B)$ constructed from $T$ by minimum augmentation. The digraph $B$ is represented by the same data structure as $B_{\text{out}}$ in the KDTB algorithm.

*Method.* DT algorithm consists of three phases.
**(Phase 1)** Preprocess: $T$ is converted into a $k$-ary tree $T'=(V',A')$ as follows:

For each vertex $v$ in $T$ such that $\text{od}_T(v)>k$, delete any $\text{od}_T(v)-k$ arcs in $\text{OA}_T(v)$. Let $Ad$ be a set of deleted arcs. Then, $T-Ad$ consists of $|Ad|+1$ $k$-ary trees and let $T_0, T_1, \ldots, T_{|Ad|}$ be the resulting $k$-ary trees. For each $i$ ($0\leq i\leq |Ad|-1$), add an arc whose tail vertex is any leaf of $T_i$ and its head vertex is the root of $T_{i+1}$. Let $Aa$ be a set of added arcs and let $T'=(V,(A-Ad)\cup Aa)$. Note that $T'$ is a $k$-ary tree.
**(Phase 2)**: Construct a $k$-B-digraph $B'=(V,A'_B)$ from $T'$ by using the KDTB algorithm.
**(Phase 3)**: Construct a digraph $B$ as follows:

$B=(V,A'_B\cup Ad)$. □

### 7.2. Correctness and computation time of DT algorithm

**Lemma 15.** *Let $B=(V_B,A_B)$ be the digraph constructed by using the DT algorithm*

*from any tree $T=(V,A)$ and any integer $k$ ($k\geq 2$). Then, $B$ is $k$-connected and $V_B=V$ and $A_B\supseteq A$ hold.*

**Proof.** As a spanning subgraph, $B$ has the digraph $B'\in BD_k$, which is $k$-connected. Therefore, $B$ is $k$-connected. It is evident that $V_B=V$ holds.

It follows from $A_B=A'_B\cup Ad$ and $A'_B\supseteq A-Ad$ that $A_B\supseteq A$ holds. □

**Theorem 10.** *Let $B=(V_B,A_B)$ be the digraph constructed by using the DT algorithm for any tree $T=(V,A)$ and any integer $k$ ($k\geq 2$). Then, the arc set $A_B-A$ is one of the solutions of $k$-VCUAP for $T$.*

**Proof.** It follows from Lemma 15 that $B$ is the $k$-connected digraph constructed from $T$ by adding arcs. Therefore, it suffices to show that $B$ is constructed by the minimum addition of arcs. In $B$, the following (i) and (ii) hold.

(i) For each vertex $v$ such that $\mathrm{od}_T(v)\leq k$, $\mathrm{od}_B(v)=k$.

(ii) For each vertex $v$ such that $\mathrm{od}_T(v)>k$, $\mathrm{od}_B(v)=\mathrm{od}_T(v)$.

From (i) and (ii),

$$|A_B-A|=\sum_{v\in V}\max(k-\mathrm{od}_T(v),0)$$

holds. Therefore, it follows from Theorem 1 that $B$ is constructed by minimum augmentation. □

**Theorem 11.** *The time-complexity of $k$-VCUAP ($k\geq 2$) for any tree $T=(V,A)$ is $\theta(k|V|)$.*

**Proof.** It is clear that Phases 1 and 3 can be executed in $O(|V|)$ time. It follows from Theorem 9 that Phase 2 can be executed in $O(k|V|)$ time. Therefore, $k$-VCUAP ($k\geq 2$) for any tree $T=(V,A)$ can be solved in $O(k|V|)$ time by the DT algorithm.

By a similar discussion to that of Theorem 9, it is shown that the lower bound is $\Omega(k|V|)$. □

## 8. Conclusions

We have described an algorithm for solving the $k$-vertex-connectivity unweighted augmentation problem ($k$-VCUAP) in $O(k|V|)$ time for every rooted directed tree and showed that it is the optimal time algorithm within a constant factor.

The problem seems very important in improving the reliability of a communication network by adding communication lines. The problem considered in this paper is restricted to that for rooted directed trees, and the future work by authors will include extension of graph classes for our consideration.

### Acknowledgement

The authors gratefully acknowledge helpful discussions with Dr. Koichi Wada. They also thank Dr. Toshiro Araki and Dr. Yoshihiro Tsujino for their useful suggestion. They are grateful to Mr. Mitsuji Ikeda and Mr. Yoshifumi Manabe.

## References

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, Data Structures And Algorithms (Addison-Wesley, Reading MA, 1983).
[2] B. Bollobás, Extremal Graph Theory (Academic Press, New York, 1978).
[3] K.P. Eswaran and R.E. Tarjan, Augmentation problems, SIAM J. Comput. 5 (4) (1976) 653–665.
[4] S. Even, Graph Algorithms (Computer Science Press, Rockville, MD, 1979).
[5] G.N. Frederickson and J.Ja'ja', Approximation algorithms for several graph augmentation problems, SIAM J. Comput. 10 (2) (1981) 270–283.
[6] M.R. Garey and D.S. Johnson, Computers and intractability - A Guide to the theory of NP-completeness (Freeman, San Francisco, CA, 1979).
[7] K. Hagihara, K. Wada, M. Ikeda, T. Masuzawa and N. Tokura, Vulnerability of a communication network with a satellite, Trans. IECEJ, J67-D (10) (1984) 1155–1162 (in Japanese).
[8] Y. Kajitani and S. Ueno, The minimum augmentation of a directed tree to a $k$-edge-connected directed graph, Tech. Rep. IECEJ, CAS83-3 (1983).
[9] T. Masuzawa, K. Hagihara and N. Tokura, On the $k$-node-connectivity augmentation problem for undirected graphs, Tech. Rep. IECEJ, AL84-11 (1984) (in Japanese).
[10] T. Masuzawa, K. Hagihara and N. Tokura, On the $k$-node-connectivity augmentation problem for directed trees, Tech. Rep. IECEJ, AL84-27 (1984) (in Japanese).
[11] T. Masuzawa, K. Hagihara, K. Wada and N. Tokura, The $k$-node-connectivity augmentation problem for directed binary trees, Trans. IECEJ, J67-D, 1 (1984) 77–84 (in Japanese) (Systems•Computers•Controlls 15 (1984) 75–83).
[12] T. Masuzawa, K. Hagihara, K. Wada and N. Tokura, The graph classes with optimal algorithms for $k$-connectivity augmentation problems, Proc. Internat. Conf. on Computers, Systems and Signal Processing (Bangalore, India, Dec. 1984).
[13] T. Masuzawa, K. Hagihara, K. Wada and N. Tokura, The optimal time algorithm for the $k$-node-connectivity augmentation problem for directed ternary trees, Trans. IECEJ, J68-D (to appear) (in Japanese).
[14] S. Ueno, Y. Kajitani and H. Wada, The minimum augmentation of trees to $k$-edge-connected graphs, Tech. Rep. IECEJ, IN83-6 (1983) (in Japanese).
[15] T. Watanabe and A. Nakamura, Vertex-connectivity augmentation problems, Tech. Rep. IECEJ, AL81-26 (1981) (in Japanese).
[16] T. Watanabe and A. Nakamura, $k$-edge-connectivity augmentation problem, Tech. Rep. IECEJ, AL83-90 (1984) (in Japanese).
[17] T. Watanabe, A. Nakamura and M. Takahashi, Augmentation problem for a vertex subset of a graph, Tech. Rep. IECEJ, AL83-89 (1984) (in Japanese).

## Appendix

### *Glossary of notations*

| Notation | Brief explanation |
|---|---|
| $\langle a_1, a_2, \ldots, a_p \rangle$ | the sequence of $a_1, a_2, \ldots, a_p$ in this order |
| For a finite set $A$ | |
| $\langle\langle A \rangle\rangle$ | any sequence which consists of all elements in $A$ |
| $\langle\langle A \rangle\rangle_i$ | the $i$-th element of $\langle\langle A \rangle\rangle$ |
| $[m \cdots n]$ | the set of integers $\{m, m+1, \ldots, n\}$ |

| | |
|---|---|
| $V(G)$ | the vertex set of $G$ |
| $A(G)$ | the arc set of $G$ |
| For an arc $e=(u, v)$ | |
| $t(e)$ | the tail vertex $u$ |
| $h(e)$ | the head vertex $v$ |
| For an arc set $A'$ ($\subseteq A(G)$) | |
| $t(A')$ | $\{t(e) \mid e \in A'\}$; the tail vertex set |
| $h(A')$ | $\{h(e) \mid e \in A'\}$; the head vertex set |
| For a vertex $v$ in $V(G)$ | |
| $\mathrm{IA}_G(v)$ | $\{e \mid h(e)=v\}$; the incoming-arc set |
| $\mathrm{OA}_G(v)$ | $\{e \mid t(e)=v\}$; the outgoing-arc set |
| $\mathrm{Fin}_G(v)$ | $t(\mathrm{IA}_G(v))$ |
| $\mathrm{Fout}_G(v)$ | $h(\mathrm{OA}_G(v))$ |
| $\mathrm{id}_G(v)$ | $\lvert\mathrm{Fin}_G(v)\rvert$; the indegree of $v$ |
| $\mathrm{od}_G(v)$ | $\lvert\mathrm{Fout}_G(v)\rvert$; the outdegree of $v$ |
| $G[V']$ | the induced subgraph of $G$ with the vertex set $V'$ |
| $G-V'$ | the digraph $G[V-V']$ |
| $G-A'$ | the digraph $(V(G), A(G)-A')$ |
| $u$–$v[V']$ path | a $u$–$v$ path in $G[V']$ |
| $(V_1, V_2)[V']$-link | a $(V_1, V_2)$-link in $G[V']$ |
| $(v, V_1)[V']$-fanout | a $(v, V_1)$-fanout in $G[V']$ |
| $(V_1, v)[V']$-fanin | a $(V_1, v)$-fanin in $G[V']$ |
| $G=(V, A, v_0)$ | the tree $G=(V, A)$ with the root $v_0$ |
| $\mathrm{depth}_G(v)$ | the depth of $v$ in the tree $G$ |
| $G(v)=(V(v), A(v))$ | the subtree of $G$ with the root $v$ |
| $n_G(v)$ | $\lvert V(G(v))\rvert$ |
| $v^i$ | the $i$-th son of $v$ |
| $P_G(u, v)$ | the maximum number of mutually internally-disjoint $u$–$v$ paths in $G$ |
| $C_G(u, v)$ | a vertex set with minimum cardinality to separate $v$ from $u$ |
| $K_n$ | the $n$-vertex complete digraph |
| $D_{n,m}$ | the $n$-vertex super daisy chain of degree $m$ |
| $\underline{BD}_m$ | the class of $m$-B-digraphs |
| $\overline{m}$ | $m \bmod \lvert V(G)\rvert$ |